SINCLAIR

Every month £1.75

JUNE 1990

4 TO 7 9 10/11 13 14 MICRO DRIVE MOD
18 20 24 26 30 38 40 42

QL WORLD

NEW SERIES

ARCHIVE POWER

"TAPPING AN UNDERUSED RESOURCE"

MEDIA MANAGER · SPECIAL EDITION

UPDATE WITH EXECUTIVE STATUS

QL SUPERBASIC

USING THE WHAT?

LDEF FOR SUPERBASIC

THE PROGS: PIPING HOT

SOFTWARE FILE-MS QLINK

ISSN 0951-9335

06

9 770951 933009

SINCLAIR QL WORLD

Editor
Helen Armstrong

Production Controller
Jayne Penfold

Designer
Jeff Gurney

Advertising Sales
Jason Newman

Magazine Services
Sheila Baker

Advertising Production
Michelle Evans

Group Advertising Manager
Richard Vaughan

Group Editor
John Taylor

Publishing Director
Ray Lewis

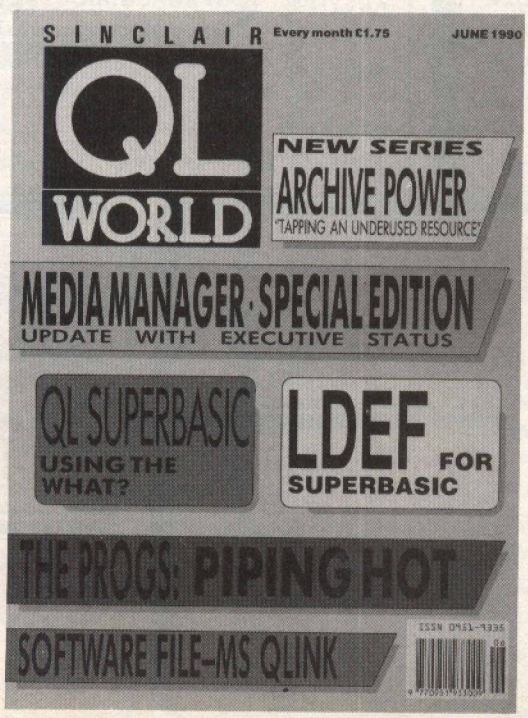
Managing Director
Peter Welham

Sinclair QL World
Panini House
116-120 Goswell Road
London EC1V 7QD
Telephone 071-490 7161
ISSN 026806X

Unfortunately, we are no longer able to answer enquiries made by telephone. If you have any comments or difficulties, please write to The Editor, Open Channel, Trouble Shooter, or Psion Solutions. We will do our best to deal with your problem in the magazine, thuough we cannot guarantee individual replies.
Back issues are available from the publisher price £2 U.K., £2.75 Europe. Overseas rates on request.
Published by Maxwell Consumer Magazines, A Division of MCPC Ltd. S M Distribution, Streatham, London SW1. 01 677 8111.
Subscription information from: MCM Subscription Dept, Lazahold Ltd., PO Box 10, Roper St, Pallion Ind. Est., Sunderland SR4 4SN. £21.00 U.K., £24.70 Europe, Middle East £25.80, Far East £27.60, Rest of World £26.20, U.S.A. $45.00. Airmail rates available on request.
Typesetting by Adtec Typographics, Britannia Court, Basildon, Essex. Tel: (0268) 591110.
Printing by Cradley Print.
Sinclair QL World is published on the fourth Wednesday preceding cover date.

CONTENTS

JUNE 1990



NEXT MONTH

SOFTWARE FILE: THE VOYAGE OF THE BEANO
A new text and graphics adventure from Alan Pemberton.

ONE MAN'S SYSTEM
Including a Hunt routine for use in Archive.

QL SCENE

THE
VOYAGE OF THE
BEANO
A
TEXT AND GRAPHICS ADVENTURE
FOR THE SINCLAIR QL
WITH 256K+ MEMORY

WRITTEN BY
ALAN PEMBERTON
GRAPHIX BY
FRANCIS O'BRIEN &
ALAN PEMBERTON

PUBLISHED BY C.G.H. SERVICES,
CWM GWEN HALL PENCADER, DYFED, CYMRU
SA39 9HA

Ahoy!

A new offering from adventure and public domain publishers CGH Services is *The Voyage of the Beano*, a text and graphics adventure for QLs with 256K-plus of memory.

The Voyage of the Beano charts the adventures of J. D. Hogwash, a lowly deck-hand and shove-ha'penny champion, who captures a Spanish galleon by accident and is dispatched to the New World by an impressed Queen Bess to fight the Spaniard and bring back the gold.

His fate is in the hands of the player.

The Voyage of the Beano is written by Alan Pemberton, with graphics by Francis O'Brien and Alan Pemberton, and is so new that *QL World* does not have a price at the time of writing. Contact **CGH Services, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA** for further details.

Multiple Basic

Minerva rom publishers QView are now making multiple Basic interpreters available with the latest versions of Minerva. This will make it possible to have multiple concurrently-running Super-Basic interpreters running on Minerva, a feature expected when the QL was originally launched, but never achieved.

A small EXECutable job will now be supplied,which can be hotkeyed using the QJump Hotkey System 2: this calls a new vector in the Minerva rom, which promotes it to copy of the Basic interpreter. When invoked, a new Basic can copy the entire Basic name table or just the original rom keywords.

Other new vectors include a fast memory move routine which, states QView, can move memory non-destructively (in the case of an overlap) at 359K per second.

With the new version comes a price increase to £40 (£35 to Quanta members) with £2.50 post and packing to overseas customers, printed documentation, no further requirement for a rom image at time of ordering, and a credit card ordering facility on 0480 412884.

For orders or information contact **QView at 29 Carnaby Close, Godmanchester, Huntingdon, Cambs PE18 8EE.**

Free Iron

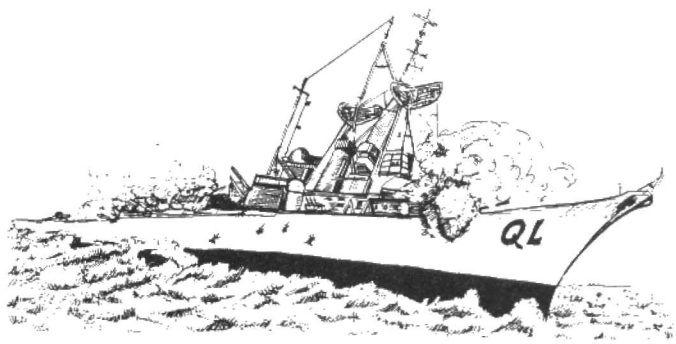
The Spring 1990 supplement to the Greenweld Electronics catalogue has 32 pages, a special introductory digital multimeter offer, a free 240V ac miniature mains 15W soldering iron with over £20-worth of orders from any current Greenweld catalogue, "amazing bargains" and a new address, which is:

Greenweld Electronics Ltd., 27 Park Road, Southampton SO1 3TB. Tel. 0703 236363.

QL ADVENTURERS'
FORUM

ISSUE 9 PRICE £1.25

THE LAST ISSUE!



H.M.S. UNSINKABLE ?
THE QL HAS SURVIVED ATTACKS FROM GREY WOLF AND TYPE 22. NOW THE BIGGEST CHALLENGE - FLEET TACTICAL COMMAND

INSIDE THIS ISSUE:

- REVIEWS OF FLEET TACTICAL COMMAND AND RETURN TO EDEN
- PLUS OUR USUAL HINTS, HELPLINE AND SOLUTIONS FOR ADVENTURES
- AND A SPECIAL SELECTION OF ARCADE GAME REVIEWS

What's in a name?

QL Adventurers' Forum (QLAF) issued 9 is now out. In a surprise move, the front cover displays the words 'The Last Issue!' The reason behind this is not the magazine's demise, but a change of name: publisher Richard Alexander, is relaunching QLAF as *QL Leisure Review*, to reflect the broader range of games, including arcade games, now appearing in its pages.

Issue 9 contains reviews of *Fleet Tactical Command* from Di-Ren, *Treasurer Hunt* (public domain) from CGH, *Brain Teaser* from Jochen Merz, *Speedfreaks* from Kaos and *Assault and Battery*, also from Kaos, *Dreamlands* from Jean-Yves Rouffiac and *Uncle Loony's Legacy* by the Watsons. Also appearing are solution hints to *Aquanaut 471* and *Colossal Cave*, news, short reviews, hints, and a wargames update.

QL Technical Review issue 3 is also out and about. Alongside a large number of brief reviews are an article on adjusting television displays, comms (mainly bulletin boards), programming hints, and a considerable amount of topical news and responses to previous comments.

QL Adventurers' Forum is £1.25, and QL Technical Review, £1.50, both published by **CGN Services, Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA. Tel. Pencader 574.**

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or just sound off about something which bothers you, write to: Open Channel, Sinclair QL World, 116/120 Goswell Road, London EC1V 7QD.

Conqueror

I am writing about the comments in last issue's *Troubleshooter*, regarding the *Conqueror* PC emulator. Like all things, it has good points and some bad ones, the main criticism being its lack of speed. It is comparatively slow but I consider it has more going for it than the article suggested.

First, I am sure there are people like myself who refuse to get rid of the QL for yet more computer hardware which is not necessarily better. Second, I did not have the money to buy an IBM PC or a clone, so the £80 for the emulator was the ideal solution.

Third, Conqueror will run all the well-behaved PC packages I have tried, including *p-System*, without which I could not complete my Open University degree this year.

On the point of speed, it is not so long ago that people were using mainframe computers; when they were used to compile and link programs, for instance, they took far longer than the emulator.

Philip Johnson,
Stoke-on-Trent,
Staffs.

Flashes

A few weeks ago I started to get multi-coloured flashes on the screen of my Microvitec Cub 653 monitor which I have had longer than I can remember, possibly about 1984-5 and which is used for some hours on most days.

By eliminating other causes I found that it was the monitor which was at fault and, as I had reason to be in Bradford, decided to call at the factory to see what could be done.

The result was that, after about an hour and a quarter, while I was allowed to be with the technician testing it, I emerged with a monitor which, for some weeks has been as good as ever. The cost was very satisfactory – less than the installation of a new membrane in the QL.

The explanation given was that some of the components had started to pass too high a voltage and the tolerance limits had been passed. I was told that the 653 is no longer made but that a one-off could be made to order at a reasonable price.

Arthur Nunn,
Ripon,
N. Yorks.

Spell

Bryan Davies is correct; text[87] could have made more of a splash about its spell-checker. It is brilliant. Perhaps it is not clear that English, French and German are all supplied as standard, although the user would normally select only one at a time.

My only slight complaint is that the dictionary editor uses a window hard up to the left-hand edge of the QL screen. As a monitor I use a television set with RGB which does not give the full width, so I have to do some guesswork. Of course, you need the editor only when you want to alter the dictionary.

On another topic, I was puzzled by your editorial remark in February that microcassettes are cheaper than discs. Davies mentioned paying between 66 and 83 pence for a disc. I pay as little as £1 each for discs guaranteed for life. If a disc ever fails to read or write then the supplier, Inmac, sends two discs as replacement. The reliability is important to me, although I could buy more cheaply. Even on a one-for-one basis, ignoring the huge capacity advantage of discs, they are much cheaper than microcassettes.

Ken Whyld,
Caistor,
Lincoln.

Editor's comment: This is my mistake. I use branded 3.5in. discs and buy by the box. Through a combination of good housekeeping and cadging the odd disc from my partner, I have not had to order another box of discs for a long time. Last time I ordered it cost me £40 for a box of 10. That will give you some idea of the time lag. The price of 3.5in. discs has fallen dramatically since then. I do much of my professional work on company-owned Amstrad PCWs and discs for them are still advertised at between £2 and £4 each, 10-off; 5.25in. discs are the cheapest

Value

Your editorial in the March, 1990 issue of *QL World* invited comment from readers concerning the relative amount of editorial matter in the magazine, compared to that obtaining, say, two years ago. I would suggest that *QL World* is still the only important source of further information on the internal arrangement and functioning of the 150,000 QLs which were sold. It is also the general source of information about spares, programs and service, for an exceedingly useful but greatly under-rated

Editor's notebook

Have you heard the one about the magazine publisher, the computer genius and the QL World editor?

One day, a Magazine Publisher, a Computer Genius and a QL World Editor — any one you like — were arguing about whose was the oldest profession.

"Mine is the oldest profession", said the Magazine Publisher. "Once the Creator finished creating the World, He needed someone to spread the News. He couldn't spend the rest of eternity with a megaphone, could He?"

"Rubbish", said the Computer Genius. "Long before you had any News to spread, my Genius was required to help bring Order from the primaeval Chaos."

"Very good," said the QL World Editor wearily, "and where do you think the Chaos came from?"

Normally, the gag would end there, but this time the Magazine Publisher and the Computer Genius (any one you like) turned to the QL World Editor and said with one voice:

"Don't give yourself airs, buster. You couldn't have done it without us."

My telephone is still in a crate. We're still running a little bit late, but we're working on it.

instrument, so that the amount of editorial matter in it is not necessarily the factor uppermost in the minds of all purchasers, though it is important to many.

Sales as high as 12,400 copies monthly surely confirm that view. They mean that one in 12, roughly, of all QL owners buy the magazine. The unknown is, of course, how many QL owners are active, in the sense of needing more than the basic 128K computer and software originally supplied, and hence, further equipment and/or servicing information.

In the absence of facts, one must assume, statistically speaking, that the figure is 50 percent, so it must be that about one in six active users are buying *QL World*. Even that is probably a pessemistic estimate, since a fair fraction of the QLs sold went overseas. This must represent continuing good acceptance and acknowledgment of the magazine by any normal commercial standards.

The number of pages is important, I suppose, to convince some people that they are getting value – but far more important is the usefulness of their content. In that respect, recent articles have seemed more and more to acknowledge that, while expensive, purchased programs in machine code are, one assumes, the ultimate luxury in trouble-free computing, SuperBasic programs can be made to do as much, even if a little more slowly, for those who are not in a panic to see the result. It is articles such as Simon Goodwin's *Taskforce* facility which make the magazine so welcome and boost its sales.

L.G.L. Unstead-Joss,
Edinburgh.

Lockup

I have keyed-in *Skyburst* from the February, 1987 issue in the belief that because it has checksums on each DATA line it would work if it showed no errors. That, however, seems not to be the case. As soon as a missile strikes one of the 'invaders' the program locks up, requiring the QL to be reset to regain control.

Two-and-a-half years later, do you have any record as to the nature of the fault? It occurs to me that it may have been written for a 128K machine and mine is expanded to 640K. Any help you can provide would be much appreciated.

Stuart Winnall,
376 Carterknowle Road,
Sheffield,
South Yorkshire S11 9GD.

Editor's comment: The author of Skyburst was Simon Ovenston, whose address is apparently no longer in the files. If he is reading this, or if anybody can shed light on the problem, could they please get in touch with the writer?

Animations

Browsing through my collection of your wonderful magazines I noticed the Screen Storage toolkit in the September, 1987 issue. I typed it in and have created some simple but fairly impressive animations with the new commands.

The animations I have done so far have required each frame to be drawn out by Basic first and then stored as machine code in memory by the G_SAVE command each time the animations are run.

That is not very convenient, as it makes the program very slow, especially with complicated vector images. I have had to do this because there is no reference to the saving of the code produced by the new commands. Therefore could you explain how to save the code produced and how to reload it so that the G_LOAD command can be used to produce the animations without having to go through the painfully slow Basic drawing of it all first?

I would also like to know if there are any video digitisers available for the QL and the address of the supplier.

G. Ayre,
Gainsborough,
Lincs.

Editor's comment: We will look into the animation question. As for the digitiser, TK Computerware was the agent for the SPEM digitiser; we are not sure of the status of this at the moment but TK may be able to advise you; see Instant Access on page 50 for the telephone number.

CritiQL

April's edition of QL World was prey to more misprunts than usual, especially on the Open Channel pages.

A letter from Christian Caris, appropriately headed 'Confused', seemed to suggest that he is in the process of having the Editor sued (sic) for serious word processing. Mr Caris should know that his litigation has no chance if Ms Armstrong uses Quill.

In *Troubleshooter*, Bryan Davies rightly takes to task certain ads for QL Organiser for not mentioning the need for a £60 interface. The advertisers say that users will become aware of that when they read the manual, but by then they will have bought the Organiser, won't they?

The computer industry is not alone in concealing the true cost of advertised goods and services ("Ex. VAT" isa favourite). Mr Davies does well to draw our attention to this.

Mike Lloyd gives us an interesting article about Desk Top Publishing, a subject which seems almost as jargon-ridden as computing. His piece is about layout and he does not mention the sheer hard work required by the reader to decipher the small print. My eyes are fairly efficient if I wear my specs, but I find most DTP literature very tiresome to read — not least the efforts of one or two of your best advertisers. Digital precision is not enough — you've got to be able to read DTP without resorting to a magnifying glass. And all DTP looks dreadful to my eyes.

No Simon Goodwin this month. Pity, especially as in March's issue he said he would return to the subject of his ingenious *Multibasic* routines. Mr Goodwin's articles are always worth reading, but his hexloader would save me a lot of work if he could find a way of distinguishing between letter 'B' and integer '8' in his hex DATA statements. Would it not be possible to print (say) 'X' for 'B' and have the loader do the necessary conversion to decimal 11? It would save a lot of 'Checksum incorrect' messages.

A similar difficulty emerges when typing-in SuperBasic programs. Some programmers expect me to type strings of spaces which, by their very nature, are invisible. Could they consider inserting REMs at the end of offending lines to give the number of spaces in their strings? Or perhaps they could try PRINT FILL$ ("",20) which will print 20 spaces with no bother at all. And we'd all be saved trying to count them.

M T Edwards' *Code Breaker* program is interesting, but I wish he had this kind of regard to 'typing-in' problems. He also defined a WINDOW#2 (line 330), which makes it impossible to correct typing errors after a RUN because the LIST command defaults to channel 2. I had to write a PROC to re-define the default screens in order to edit his program. Not a difficult task, but he could have used another channel for his WINDOW#2.

I'm going to start a Society for the Encouragement of User Friendly Listings, with severe penalties for those who transgress.

Finally, let me attempt to answer K. Plummer's printing problem which is posed in Open Channel. Could it be that he is forgetting to ENTER 'BAUD 1200' (or whatever his baud rate happens to be) before 'OPEN#3,ser1' and 'LIST#3'? I am always omitting this.

Even when I remember the baud rate, I forget to switch the printer on!

C.B. Storey
Tyne & Wear

How wise your suggestions are, CB, and how kind of you to blame the 8s and Bs on Simon's cruel printer rather than my own. Perhaps a magnifying glass is the best answer. We can't print them much larger without larger pages.

Advertisers should not expect their customers to read the manual (before or after purchase). It is true that the manual may hold a wealth of valuable information (as well as revelations about expensive add-ons, but why fly in the face of experience?

My imminent suage would appear to be a perfect example of computer-checked English: it makes perfect sense, as long as you don't know what it means.

QL SCENE

Microdrive stocks saved?

EEC Ltd., suppliers of new boxed QLs, QL hardware, microdrives and cassettes and disc drives to the QL and Spectrum community, have contacted *QL World* to say that they have "plenty" of QL and Spectrum Plus 2 and Plus 3 microdrive cartridges, "and foresee adequate stock of hardware for at least 12 months, and spares for two to three years, including Microdrive cartridges."

Bill Richardson, EEC's managing director, says "The QL demand is still good. We have sold over 400 machines in the last twelve months and have plenty in stock. These are still new machines, brought up to date with roms and software. We have large stocks of Microdrive cartridges to support our customers and confidently expect continuing supplies to become available. We also have lots of spares."

EEC has pledged its continuing support for QL World and Quanta, and appeals to QL World in its turn to continue to support QL users. EEC is offering a free one-year membership of Quanta to any customer spending over £100 with EEC. Bill Richardson has also spoken to QL Microcassette manufacturer Ablex's MD, Peter Banks, recently. Richardson believes that there is a very substantial stock of mechanical parts for the cartridges in existance, and that supplies of suitable tape should become available in May or June.

At Ablex, Peter Banks confirmed that he had located supplies of tape at German tape giant, BasF, similar to that previously in use in the Microcassettes. He had samples already in hand and production tests were due to begin in late April.

The high-quality video tape used to construct the continuous-loop Microcassette tape is unusual in that it is neither lubricated in the normal manner of continuous loop tape, nor has the coated backing used by most videotape to give reliable tracking characterisitics. "We and Sinclair investigated the original tape requirements together and found that videotape had sufficient record quality but the normal lubrication or coated backing would foul the microdrive tape heads in time, so we had to find a tape with a shiny backing."

Sample tape with shiny backing is now with Ablex.

QL in Italian

QL Italy is a public domain QL disk magazine published in Italian. The disk will run on any QL with at least one 720K disk drive and is "packed full" of QL-related items. Nine issues have been published since February 1989, and QL Italy is aiming for bi-monthly publication. Typical contents include reviews, articles, programming information, small programs, demos and utilities, letters and other comments. Scanned screen shots are taken from commercial programs to give the prospective buyer a look at what the program offers.

QL Italy is available for the cost of the disk and postage: 5,000 Lira in Italy, 6,000 Lira (a bit under £3) in the EEC; users living outside the EEC should send a stamped self-addressed envelope or IRCs for information. The address for orders and information is **QL Italy Group, c/o Eros Forenzi, via Valeriana 44, 23010 Berbenno (Sondrio), Italy. Tel. (Italian local) 0342 492323.**

News-Ware

Another publication available in Italy is *News-Ware*, the bulletin of Club-Ware, established as a QL club in Italy since 1987. News-Ware is a bi-monthly magazine composed using *text*^{87}, containing news, reviews and articles on specific topics. Membership of Club-Ware gives access to the newsletter, support advice and a public domain software library. Annual membership costs 25,000 Lira, paid to **Club-Ware, c/o Roberto Orlandi, via Brescia 26, 25039 Travagliato (Brescia), Italy. Tel. (Italian local) 030 6863311.**

Enthusiasts hunting for bargains at the All Formats Computer Fair.

Fun of the fair

The second All Formats Computer Fair on April 28 and 29 was arranged in 'villages' for many of the more popular specialist computers, including the QL, Z88 and SAM Coupe. The following fair, which is to be held on June 9 and 10, will have Archimedes and MSX villages.

Like the Z88, the Archimedes appeared on the market in the same generation as the Atari ST but failed to make the impression which the ST and before it the IBM PC made on the home and small business computing market.

The electronic magazine *Micronet* is expected to be on line at both fairs. The site is again expected to be the New Horticultural Halls, near Victoria Station, London. Phone **Bruce Everiss on 0926 640137** for further information.

OPD TO QL

Dennis Briggs outlines the connections to reuse One Per Desk microdrives.

One of the long-term problems with the QL is the reliability of the Microdrive units. Before owners tell me that theirs are satisfactory, consider the fact that programs have appeared which will make back-ups of protected cartridges so that the program is not consigned to the dustbin if the Microdrive will not read it.

One of the main reasons for failure is the extremely hot heatsink next to the Microdrive ULA which causes it to lose performance over a period of time and read/write heads which wear out.

The Microdrive units can be changed or repaired at a price; also many owners with disc drives just want the facility to read or write to a tape on rare occasions. This an omission on the Thor XVI.

ICL manufactured One Per Desk or Tonto computers used Microdrives together with the 68008 chip and the two QL ULAs but very little else. Those OPD modules containing two Microdrives are available for about £5 per module, so it is worthwhile considering fitting them as a plug-in expansion to the QL.

Top overcome any problems we have used *Page Designer 2* and a professional artist to produce the connections, a stylised PCB overlay and a double-sized PCB overlay. With a little good fortune the PCB overlay should be printed actual size.

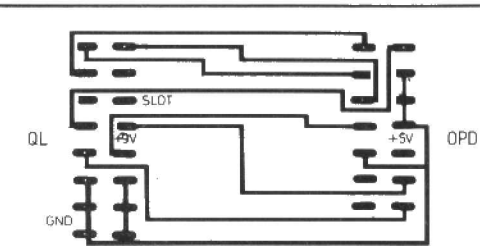


Figure 3. Stylised connection diagram.

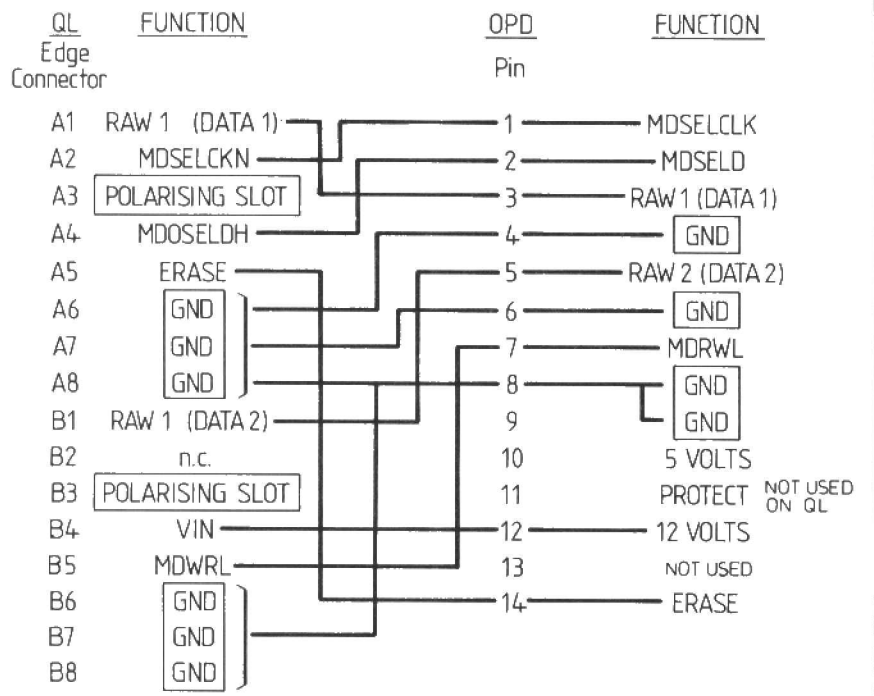


Figure 1. Connections between the QL and One Per Desk.

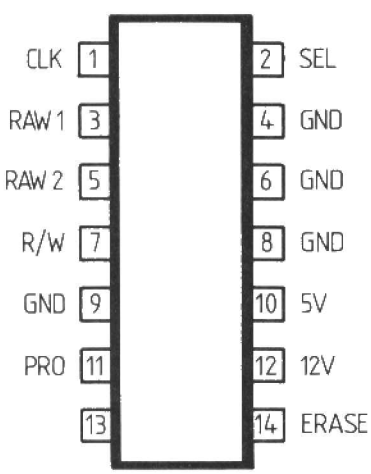


Figure 2. One Per Desk pin out.

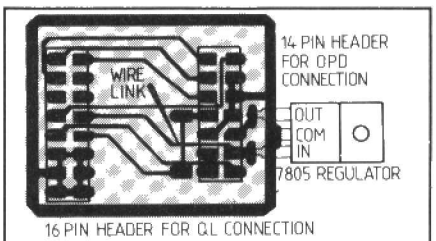


Figure 4. The connection overlay.

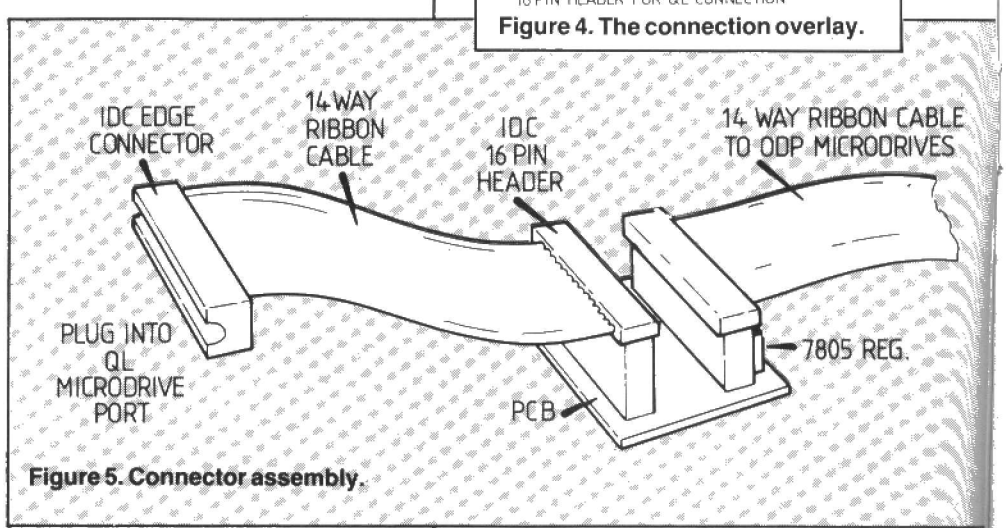


Figure 5. Connector assembly.

TROUBLE

APROBL

Improvements in the Trump Card and other news about Miracle Systems, and an introduction to some good new programs from Bryan Davies. News from Troubleshooter's investigations into readers' commercial queries is less encouraging this month.

Miracle Systems have developed an adapter to enable users of older versions of the Trump Card to connect three or four floppy disc drives to their systems (the previous limit was two). The adapter plugs into the output connector on the Trump Card and has two similar connectors at its output end. Standard one- or two-drive ribbon cables can be plugged into both connectors. The unit is about 5x 5cm. A new rom chip is supplied for the Trump Card. The price is £15. With the advent of PC emulators, and a variety of file transfer programs, some users will certainly appreciate being able to have separate pairs of 5¼in and 3½in drives connected to their system at the same time.

The current version of the Trump Card is different in several ways from the earlier one. It does support four floppy drives directly but they must be daisy-chained to the one output connector. The memory chips fitted are 80ns, 1MB dram types (but maximum capacity is still 769KB), with the result that there is a lot of spare space on the pcb, and memory accesses are now much faster; the TC should now rank with the old CST interface, having memory access as fast as it can be on the QL. Another advantage of the new chips is lower power consumption; this, along with some redesign of the circuitry, should result in few problems with lockups. The new feature likely to appeal most to prospective purchases is the drop in price to £225 for the full 768KB version.

No high density

In answer to some possible questions, Miracle state that they considered developing a high-density drive interface, but have — for the present — dropped the idea. Such an interface would have required a different drive chip from the WD1772 used in the Trump Card and, presumably, the driver software would have to be rewritten to cope with 1.2/1.44 MB. As regards producing a hard disc interface on its own, for users who prefer to provide their own drive, Miracle feel that the potential problems far outweigh any possible profit; that is, the letters and calls they would get from users experiencing difficulties getting their drives to work would take far too much time to deal with. With their present policy of supplying only the complete hard drive system, Miracle point out that users have a clear guarantee of support, whereas the only support for an interface alone would be its replacement in the event of it being defective. In view of the variety of hard drives for sale, and the uncertain state of software which could be used to drive them, this view seems very sensible.

Tandon drives

Potential purchasers of the hard disc system who have noted the adverse comments about Miniscribe, the hard disc manufacturers, will be glad to know that Miracle use Tandon drives, which are produced by the drive-making arm of Western Digital, a well-known drive controller and chip company. Miniscribe looks like continuing as part of the Maxtor drive company. The speed with which the drive feeds the screen with a series of 32KB screen dumps in a demonstration routine used by Miracle is certainly impressive.

> "The Miracle Systems adaptor enables users of older versions of the Trump Card to connect three or four floppy disc drives to their systems . . ."

The Midi music interface that was supplied briefly by Miracle is thought to be undergoing a software revamp and may be made available again.

Software[87] have been developing an Italian version of *text*[87], and it should be available about now. Not surprisingly, the French and German versions took a little longer to complete than the English version, and *Software*[87] apologise to any overseas buyers who experienced delays. Potential buyers who are still hesitating because of the adverse comments about the early version of the program should talk to people with the later versions, because the user interface is now good, and reliable, and the latest manual is much more comprehensive. Incidentally, in case any readers are confused about Terry Harman's involvement with *Software*[87], he has no direct connection with them at all, but does offer an independent help line service for *text*[87]. Users who feel they need assistance in learning how to use *text*[87] can contact Terry and negotiate a support deal with him.

The small Portuguese supplier Qfile is offering a program for transferring files between MS-DOS and QDOS, for the princely sum of £12. *MS-QLink 1.3* provides a variety of functions in addition to file transfer, such as formatting MS-DOS and QDOS discs, erasing files, copying complete discs either way between the two formats, and directories of both formats. A review of the program should have appeared by this time.

A review of *Media Manager Special Edition* should be appearing. And it won't be a short one. By and large, I'd steered clear of investigating problems with discs and cartridges for the past year or two, because the subject seemed too complicated to be covered in the relatively small amount of time I was willing to devote to it. On the other hand, necessity forced me to understand and make use of disc utilities on the PC; to be honest, those programs were much easier to use than the original *Media Manager*, which was the only

SHOOTER

E M S O L V E D

obvious program on offer for the QL. MMSE has altered the situation. The considerable time spent with the program to write about it, and the much-improved user interface, have revealed the program as a pleasure to use, rather than a fight, as was the case with the initial version. It is not now necessary to have a degree in assembly language programming and electronics theory to be able to retrieve

> "Some of the more enterprising smaller suppliers are looking for buyers of their lines, and there may be changes of ownership before long . . ."

lost files. It is also quite interesting to use the program just to see how information is stored on disc or cartridge.

Readers who feel strongly enough about things they would like to see incorporated in the magazine should *write now*. This is a time when the approach to the magazine is being reconsidered, and we are being asked to make suggestions about improving the format and content.

Court action

There has been a report that the S.U.B. Post Office box number was "out of action", but a subsequent check revealed that it was still in use. However, Miracle systems have now obtained a High Court judgement against S.U.B. for a substantial sum of money, and it is possible that this will force S.U.B. to suspend operations completely. **Manolis Christodoulou** wrote from Greece, to say that he ordered a SPEM kit and keyboard from S.U.B. in September 1988 and had not received it as of February 1990. There seems little doubt that he will not now get what he ordered; it seems unlikely that the SPEM company would supply goods to S.U.B. even if the latter are dealing with outstanding orders. I can only suggest that Christodoulou writes to the local Durham Trading Standards Officer (see **Information**) and gives details of his complaint. On a more positive note, **Melvyn Hiller** has received two postal orders in refund for the A4 paper not supplied to him; he had written threatening Small Claims Court action if no refund was made.

Market search

Some of the more enterprising smaller suppliers on the QL scene are looking for buyers of their product lines and it is possible there will be announcements of change of ownership of certain products before long. Most of the products on offer are thought to be insufficiently developed as they currently stand, so there may be delays of some months before they reappear, in improved form. However, at least two major titles are expected to be handled by a different supplier shortly.

Jeff Wass wrote (mid February) to say he has been unable to obtain the 2.38 version of *Archive* and the Psion printer installer from **PDQL**, despite supplying a disc, and making several 'phone calls to enquire about the orders placed in November '89. There have been other similar complaints in past months, concerning slowness in dealing with small orders, but no complaints concerning orders for more expensive goods.

J.P.S. Pennycook asked for the details of connection used by **John Acielo** in his "boxed QL" (see *One Man's System* in the January 1990 issue of *QL World*). The table that should have appeared with the article is this one:

SLOT LINE-->	SP0	SP1	SP2	SP3
0	0	0	0	0
1	0	0	0	1
2	0	0	0	1
. . . and so on, to . . .				
15	1	1	1	1

The "O" indicates 0 volts (ground/earth connection) and "1" indicates +5 volts in-series with a 1kohm resistor. **Adman Services** can supply an unbuffered 64-way adapter for the QL expansion port, with a variable number of output connectors to suit users' requirements.

Norman A. Larkin reports having problems with a PS/2 keyboard purchased rom **Schön** in the middle of last year. He says he has not had a working QL for ten months, since getting the keyboard. Numerous calls to the suppliers have produced neither a refund nor a working system. As of early March this year, the keyboard and QL have apparently been with Schön. My information is that Schön have been generally uncontactable for the past few months and have not been actively trading. As in all such cases, the options open to the unfortunate purchaser are few: a letter sent by a solicitor, demanding return of the goods or a refund within a short period, is worth trying. The local Trading Standards Officer for the district the supplier trades from (see **Information**) should be contacted, and given all details of the complaint. A claim can be made through a Small Claims or County Court, but the court may have to be in the supplier's area and that is far from where Mr Larkin lives (see **Information** for suggested Court to contact). This is however worth making enquiries about at his local CAB or county court. The expenditure on such actions could run up to about £50, plus a certain amount of time. Costs should be recoverable for a simple court action. The more complicated the complaint, the more complicated it is to recover costs.

Information

Hard disc system £449, 4-disc adapter £15, new Trump Card £225: Miracle Systems, 25 Broughton Way, Osbaldwick, York YO1 3BG
Tel: 0904 423986

***text*[87] v3.00 English/Italian/German/French versions £60:** Software[87], 33 Savernake Road, London NW3 2JU.

***text*[87] *independent* support service, including tutorial disc:** Terry Harman
Tel: 0604 842875

MS-QLink £12 (5¼" or 3½" disc only payment by sterling cheque): Qfile, Apartado 2110, 1103 Lisboa Codex, Portugal.

Unbuffered 64-way adapter: Adman Services, 53 Gilpin Road, Admaston, Telford, Shropshire TF5 0BG.
Tel: 0952 255895.

Trading Standards Officer for Surrey: P.G. Harris, County Trading Standards Officer, Trading Standards Department, Mount Hill, South Street, Epsom, Surrey KT18 7PT. Tel: 03727 40401/9.

County Court for Surrey: The Chief Clerk, Epsom County Court, The Parade, Epsom, Surrey KT18 5DN.

Trading Standards Officer for Durham (South): M. Horner, Divisional Trading Standards Officer, Consumer Protection Department, Town Hall, Darlington, Co. Durham DL1 5QY.
Tel: 0325 380651.

SUPER BASIC

When do we get a WHEN that works in SuperBasic? Mike Lloyd answers this often-asked question.

Of all the *SuperBasic* topics the one which perhaps generates the most enquiries is the WHEN keyword. For a large number of QL owners their first enquiry is: "When did SuperBasic include a WHEN that worked?," because up until the JS and MG roms WHEN was not implemented at all, even though it was a reserved keyword. Even in JS roms, which must now be the most commonly-used variant in the UK, the WHEN construct does not work reliably or safely. Its less anti-social habits include reporting a "bad name", interpreting the WHEN block twice and refusing to cancel the WHEN condition. At its most virulent, WHEN can lock up the QL to the detriment of your program and ultimately your sanity.

Given its unwelcome reputation, how can WHEN be used with any degree of safety in SuperBasic programs? For owners of Tony Tebby's *Toolkit* and for the increasing number of QL users who have migrated to the Minerva eprom the answer has already been implemented because both products rewrite parts of the incomplete WHEN code. Minerva and Super Toolkit2 are completely compatible with each other, which is just as well because they are very different products with complementary roles to play. Both solutions provide a valuable addition to the SuperBasic language.

Readers with JS or MG roms but without Super Toolkit2 can experiment with WHEN processing but the results are likely to bring them grief. The main problem lies in escaping from a WHEN clause after it has been triggered. The END WHEN statement does not clear the error processing flag so that everything done after an error is considered by the interpreter to be part of the WHEN structure. As soon as a second error is spotted the computer is likely to hang or respond unexpectedly to statements typed into the Command Window.

With Super Toolkit or Minerva in place all these problems disappear, allowing the WHEN keyword to be used in two distinctly different ways. In the first format, WHEN is followed by a condition involving a variable, rather like an IF statement. No matter where in the program the variable value is changed the interpreter will see if the WHEN expression has become true. If it is true then the commands following the WHEN line are executed until the END WHEN statement is met, following which control returns to the statement following the one which prompted the jump.

An example of the first type of WHEN structure is:

```
100 WHEN print_line > 22
110    CLS: print_line = 0
120    page = + 1
130 END WHEN
```

Exactly the same functionality could be obtained using a GOSUB statement after every statement in which the variable print_position is changed, rather like:

```
200 print_line = print_line + 1
210 GOSUB 5000
. . .
5000 IF print_line > 22 THEN
5010 . . . etc etc
5090 RETURN
```

The GOSUB solution is clumsy, slow and unattractive. With the WHEN keyword you can safety forget about including GOSUB statements (which you will know by now are rather disliked by this columnist) and writing extensive subroutines. Programs can contain many WHEN clauses of this type and all can be active simultaneously, although there may be problems if more than 20 clauses are defined at once.

The most important rule for using WHEN clauses is that the WHEN block must be scanned by the interpreter during the running of a program before it can respond to any change in a variable value. The interpreter sets markers on first

1. FILE-RELATED SINCLAIR QL ERROR CODES

VAL	*CODE*	*MESSAGE, MEANING and CURE*
-7	ERR_NF	**Not found** - a SuperBasic device, usually a drive, cannot be found (eg **DIR mdv9_**). CURE: Request new drive name and try again.
-8	ERR_EX	***Already exists*** - caused by trying to create a file using a filename already used on the disk or microdrive. CURE: request new filename and try again.
-9	ERR_IU	***In use*** - caused by trying to access a file which is already linked to another channel, eg **OPEN_IN#5, FLP1_FILE: OPEN#7, FLP1_FILE.** CURE: request new filename and try again.
-10	ERR_EF	***End of file*** - caused by trying to obtain input from beyond the end of a file. (See Figure 3 for another cause.) CURE: exit data-reading loop, close file.
-11	ERR_DF	***Drive full*** - caused by trying to write to a microdrive or disk which has insufficient room for the new data. CURE 1, if you are creating a new file: Delete partially-complete file, request

reading a WHEN clause but it does not carry out the commands within the WHEN structure at that stage. Only when the WHEN condition becomes true after the clause has been scanned does the interpreter action the statements between the WHEN statement and the END WHEN statement. It is therefore usual to place WHEN clauses near to the beginning of programs.

There is no rule to prevent WHEN blocks from being declared within a procedure or function, or for the "WHEN expression" variant to call procedures or functions from within the body of the structure. The following arrangements would be considered good programming practice:

1. Placing "WHEN expression" statements in the main body of the program before any procedures or functions are defined.
2. Placing "WHEN expression" statements in an initialising procedure called early in the program's run.
3. Placing a "WHEN expression" statement in a function or procedure definition in which it is uniquely used.
4. Placing one or more WHEN statements in a subroutine (if you really must use them).

Careless programming style can lead to problems when developing or amending programs and, at worst, might jeopardise the program during runtime. Things to avoid when using either of the WHEN variants include:

1. Nesting WHEN statements inside each other or inside FOR..NEXT or REPEAT loops.
2. Placing a WHEN statement in one procedure or function definition when the variable which activates it is amended only by another, unrelated, definition.
3. Placing WHEN structures in "dead" program areas which the interpreter never visits.

The conditional expression in a WHEN statement can be any valid SuperBasic expression, such as:

```
WHEN x = 34
WHEN y > 17
WHEN z$ = "flp1_"
WHEN h > 4 OR h < 2
WHEN Y = 12 AND an$ = "Y"
WHEN xyz > abc * def
```

The expression syntax is exactly that allowed in IF statements. As the last example demonstrates, variables can appear on both sides of the equation and the current variable values are used to calculate the truth of the expression. However, all variables appearing on the right hand side of the expression must have some value assigned to them before the WHEN clause is encountered by the interpreter, ie:

		new medium, try again. CURE 2, if you are appending to a file: Close file. If possible read whole file into memory, trim any incomplete line, request new medium and save afresh.
-12	ERR_BN	***Bad name*** - caused by an invalid filename. (See Figure 4 for another cause.) CURE: request new filename and try again.
-16	ERR_FE	***Bad or changed medium*** - either a medium has been removed with files still open or the medium has become corrupted. CURE: request insert medium, try again. If process fails again, report error.
-20	ERR_RO	***Read only*** - caused by trying to write to or delete from a read-only disk or mdv. CURE: request medium be adjusted or replaced to allow files to be written.

2. CHANNEL-RELATED SINCLAIR QL ERROR CODES

(Primarily for non-file-related channels, such as windows and pipes)

VAL	*CODE*	*MESSAGE, MEANING and CURE*
-5	ERR_BO	***Buffer full*** - caused by reading too many characters into an I/O buffer without an 'end-of-line' code. CURE: remove characters from head of buffer (eg by **LET input$ = INKEY$(#7)**) before resuming input.
-6	ERR_NO	***Channel not open*** - a reference was made to a non-existant channel, or (rarely) a channel could not be opened. CURE: alter program to remove incorrect channel reference.
-9	ERR_IU	***In use*** - caused by trying to access a file which is already linked to another channel, eg **OPEN_IN#5, FLP1_FILE:** **OPEN#7, FLP1_FILE.** CURE: alter program to remove conflict.
-13	ERR_TE	***Xmit error*** - a transmission error has been discovered affecting SER1 or SER2. CURE: try again a few times, then abort and report error.
-15	ERR_BP	***Bad parameter*** - can occur when trying to read data from a write-only device such as a screen window. CURE: change program to remove error.

3. GENERAL RUN-TIME SINCLAIR QL ERROR CODES

VAL	*CODE*	*MESSAGE and MEANING and CURE*
-1	ERR_NC	***Not complete*** - usually occurs if CTRL-SPACE has been pressed. To avoid obvious problems, CTRL-SPACE cannot be trapped by a WHEN clause.
-3	ERR_OM	***Out of memory*** - no more RAM space. CURE: if possible, remove unwanted programs, large arrays, long pipes or RAM disks, but in practise this error is usually fatal.

```
100 WHEN ERRor
105   IF ERR_XP
110     PRINT#0;ERNUM!!: REPORT
115     PRINT#0; "Try again."
120     RETRY ERLIN
125   ELSE
130     PRINT#0;"Unexpected error (";
135     PRINT#0; ERNUM; ") ";
140     PRINT#0; "on line "; ERLIN
145     STOP
150   END IF
155 END WHEN
160 :
170 :
200 REMark demonstration
205 CLS: PRINT "Type in a number"
210 PRINT "(Try invalid input too)"
215 FOR x = 1 TO 5
220    AT 10,5:CLS 3:INPUT num
225    PRINT x;"/5 correct"
230 END FOR x
235 STOP
```

```
100 LET Xpos = 0
110 WHEN WinWidth > 512 - Xpos
120  WinWidth = 512 - Xpos
130 END WHEN
```

Unfortunately, the value of the variable on the left hand side of a WHEN expression can be changed without triggering the WHEN test by passing it to a procedure definition, such as:

```
200 Double WinWidth
210 DEFine PROcedure Double (WW)
220  WW = WW * 2
230 END DEFine
```

This feature is best considered to be a bug because it does not conform to the rule that SuperBasic parameters are passed by reference, in other words that any change made to a formal parameter inside a user definition affects the equivalent actual parameter outside the definition. In the example above,no matter what value is assigned to WW, and hence to WinWidth, the WHEN clause will not be triggered.

As with SuperBasic's other low-level structures, single-line WHEN statements are permitted in which the END WHEN statement is optional. A useful addition to the WHEN syntax allows WHEN processing to be cancelled by following the WHEN keyword with the appropriate variable name, so that:

```
WHEN print_line
```

will cancel all WHEN processing involving the variable print_line.

The "WHEN" variable = expression" construct is best treated as a powerful type of subroutine, and subroutines do not mix easily with the hierarchial structures provided by the DEFine PROcedure and DEFine FUNction statements. Care should always to taken to ensure that the effects of a WHEN clause being triggered are predictable and controllable. My own preference is not to use a "WHEN expression" clause unless there is no practical alternative, which means that they very rarely occur in my programs.

The second version of the WHEN clause opens up much more spectacular possibilities because it allows programmers to trap, check and respond to errors without bringing a SuperBasic program to a premature and unwelcome halt. With this WHEN variant, any error occuring after the clause has been scanned will cause the interpreter to action the statements within the WHEN block.

The basic syntax of the second WHEN variant is:

```
100 WHEN ERRor
110   PRINT #0, "Whoops!"
120 END WHEN
```

The main body can, of course, comprise any number of valid SuperBasic statements.

There are two important differences between the "WHEN ERRor" variants. Firstly, although programs can have more than one WHEN ERRor clause only the one most recently scanned will be actioned in the event of an error occuring. Secondly, Tony Tebby reports that it is unwise to call user-defined procedures and functions from within a "WHEN ERRor" clause.

To help with error processing, a number of extra SuperBasic keywords have been implemented. Over 20 functions have been declared, each one relating to one of the possible QDOS error conditions. Unlike all other functions in SuperBasic, the error code functions are not followed by brackets. To use them, something of the way in which QDOS handles errors must be known.

After interpreting every statement, QDOS issues an exit value which is ignored if the statement is actioned successfully. If, however, an error is detected the exit value is set to a negative number which identifies the sort of fault which QDOS has detected. These values can be seen, together with the function name and the error messages which they produce, in the accompanying tables.

The **tables** are based on groups of errors which are likely to occur in given circumstances so that a small number of separate WHEN ERRor routines can be written to cope with file-related, channel-related and general errors as appropriate. Some of the error codes appear twice because Sinclair Research gave them two distinct meanings. The number of possible error messages was thus reduced to an absolute minimum, but error analysis has been made more difficult as a result.

The error code functions return a one if they match an error which has occured and a zero if they do not. They can therefore be used to take action appropriate to particular circumstances, for instance:

```
100 WHEN ERRor
110 IF ERR_NF
120   PRINT "File not found"
130   PRINT "Enter new filename"
140   INPUT file$
150 END IF
160 END WHEN
```

Additionally, two other functions have been included which can be used to determine the error number and the line on which the error occurred:

```
100 WHEN ERRor
110   PRINT "Error = "; ERNUM
120   PRINT "On Line "; ERLIN
130 END WHEN
```

SuperBasic programmers should be familiar with the CONTINUE and RETRY commands which are normally typed directly from the command line during program debugging. With the implementation of WHEN ERRor blocks these commands can be used within programs to control what the interpreter does immediately after error processing.

Both RETRY and CONTINUE are analogous to the RETURN statement in standard user-defined structures and subroutines, but whereas CONTINUE sends the interpreter to the statement following the one which caused the error RETRY forces the statement to be interpreted again. Of course, there is little point in this if the result is another error, but if

the mistake has been corrected by the WHEN ERRor block then all should be well.

Super Toolkit 2 users can optionally follow both RETRY and CONTINUE with a line number at which the interpreter will resume. Tony Tebby describes this as useful, but there are obvious pitfalls in jumping out of a procedure definition because of an error and then restarting the program at some other, unrelated, point. The main value of this feature is that an error might occur in the last statement on a line and to recover correctly all the statements on that line must be repeated. The command:

RETRY ERLIN

is a straightforward way of repeating all of the statements on the line rather than just the statement which caused the error.

Finally, the REPORT command allows errors to be reported to a given channel, as follows:

```
200 WHEN ERRor
210   REPORT#3
220 ENDWHEN
```

The REPORT keyword can be followed by a channel number (the default is the command window) and by an error number. For example,

REPORT #2, -12

prints the text "bad name" in the listings window.

Owners of the *Turbo Toolkit* have a couple of alternative options with the DEVICE_STATUS and WHEN_ERROR functions, the latter only functioning in Turbocharged programs. The advantage of using the Turbo Toolkit extensions is that, should your program see commercial success, it will work with whatever version of QL rom your customers own because all the necessary code is contained in the Toolkit runtime module.

Errors which occur with a WHEN ERRor block can sometimes have unexpected consequences, especially if you have called a procedure or function from within the WHEN clause. It is therefore wise to keep "WHEN ERRor" structures short and simple to avoid problems which will inevitably be difficult to analyse and put right.

Used correctly, WHEN clauses will not only provide better error-tapping for file-access routines but they can substantially shorthen input routines. Readers with very long memories will recall that one of the first *Better Basic* articles was devoted to a lengthy routine to error-trap the input of numbers. A substantial amount of code was devoted to discarding incorrect keypresses, coping with the complexities of unary minus and decimal points, and so on. The equivalent routine using a WHEN ERRor clause is shorter and easier to understand, as the accompanying listing demonstrates.

-14	ERR_FF	***Format failed*** - the depressing news that a microdrive cannot be formatted. Disk format failures are very rare. The most common cause of this error is simply failing to put a medium in the drive. CURE: ask for new medium, try again.
-17	ERR_XP	**Error in expression** - for whatever reason, SuperBasic cannot make sense of an expression. Multiplying text strings together and inputting text into a numeric variable are two ways of causing this error. CURE: ask for input again.
-18	ERR_OV	***Overflow*** - caused by wandering away from the world of real numbers, dividing by zero, looking for the square root of a negative number, and so on. PREVENTION: check variable values prior to divisions and square roots. If necessary, ask for new variable value or replace wrong value with innocuous default value.

4. PREVENTABLE SINCLAIR QL ERROR CODES

(Prevent these occurrences during program development)

VAL	*CODE*	*MESSAGE and MEANING*
-2	ERR_NJ	***Invalid job*** - a multi-tasking call has been made to a non-existent or incorrect task.
-4	ERR_OR	***Out of range*** - caused by: - printing outside a screen window (eg **AT 56,95**) OR - referring to a non-existant array element (eg **DIM A$(5): PRINT A$(9)**)
-7	ERR_NF	***Not found*** - a SuperBasic device cannot be found, (eg **DIR mdv9_**) or a call to a non-existant procedure or function has been made.
-10	ERR_EF	***End of file*** - can be caused if a READ statement is actioned when all of the DATA statements have already been read.
-12	ERR_BN	***Bad name*** - a SuperBasic name has been used out of context.
-15	ERR_BP	***Bad parameter*** - the wrong number or wrong type of parameters have been passed to a SuperBasic procedure or function, whether part of the language or user-defined.
-19	ERR_NI	***Not implemented*** - reveals that you have stumbled upon an incomplete part of SuperBasic.
-21	ERR_BL	***Bad line*** - caused by any SuperBasic syntax error which cannot be more exactly described.

LDEF FOR SUPERBASIC

Gavin Monk explains how QDos stores functions and procedures and produces a routine to list procedures by name regardless of the line numbering.

The ability to define named procedures and functions from within SuperBasic means that GOSUB need never be used. However, this means that the actual line numbers, at which routines start, are easily forgotten. Also if RENUMber is used then it becomes impossible to know the starting line numbers of your routines. This makes SuperBasic program development difficult and cumbersome. Should you need to change or examine a routine, then you are left with no choice but to list the program and search for your routine.

This routine ends all this searching by adding the function 'LDEF' to SuperBasic which returns the start line number of a given procedure or function, allowing procedures and functions to be listed by name, eg LIST LDEF ('help') TO will list from the start of the procedure or function, named help onwards. LIST LDEF ('help') TO LDEF ('help') + 100 will list the first 100 lines of the same procedure or function.

A full explanation of the way that Qdos stores procedure and function details is given and this will allow assembler programmers to examine the built in SuperBasic routines in the QLs ROM, so that they can be used from assembler routines or copied into RAM and modified at will.

Those of you with assemblers should type in the assembler listing and save the object code produced under the name LDEF_BIN file on mdv1_. Once created the extension can be set into use at any time by entering the following: a=RESPR (200) : LBYTES mdv1_LDEF_BIN,a : CALL a.

The QL Name Table stores details of almost everthing to do with a SuperBasic program and it is constantly updated as your programs are entered, amended, loaded, and run. Details of the table are detailed below. The table starts at BV.NTBAS ($18(A6)) and ends at BV.NTP ($1C(A6)). Each entry in the table consists of eight bytes as follows:

word	one of	$0001	string variable (undefined)
		$0101	string expression
		$0201	string variable
		$0301	string array
		$0501	SuperBasic string function
		$0002	floating point number (undefined)
		$0201	floating point expression
		$0202	floating point number
		$0302	floating point array
		$0502	SuperBasic floating point function
		$0003	integer (undefined)
		$0103	integer expression
		$0203	integer
		$0303	integer array
		$0503	SuperBasic integer function
		$0300	internal use (substring)
		$0400	SuperBasic procedure
		$0602	REPeat loop name
		$0702	FOR loop counter
		$0800	machine code procedure
		$0900	machine code function
word		pointer to entry in Name List (−1 if expression)	
long word		offset into variable area if entry is a variable or negative if variable undefined. High word is start line number of DEF statement for SuperBasic procedures and functions. Long word is the absolute address of the routine for machine code procedures and functions.	

The Name List contains a list of the ASCII for the characters which make up the names. Each entry starts with a byte which defines the length of the name followed by the ASCII for that name. This table starts at BV.NLBAS ($20(A6)) and ends at BV.NLP ($24(A6)).

The best way of explaining how to use the Name Table and the Name List two tables is to work through the given example.

Firstly, all the required constants are defined and the function 'LDEF' is linked in. Secondly, the passed parameters are put onto the maths stack as strings and then a check is made to make sure that only one parameter (the procedure or function name) has been passed. If the number of parameters passed is not equal to one, the error 'bad parameter' is given.

Now the actual guts of the function begin. The program looks through the Name List until it finds an entry with the same length as the string entered. If an entry of equal length is found then the routine COODBE is jumped to, otherwise the search continues until the end of the Name Table is reached and then the error 'not found' is given. The COODBE routine checks through the Ascii for the name and compares it with the Ascii for the

```
1 REMark ************************************************************
2 REMark * Basic Loader to add the function LDEF to SuperBASIC which      *
3 REMark * returns the line number of the given procedure or function     *
4 REMark *                                                                *
5 REMark * Copyright (c) G.B.Monk                                        *
6 REMark ************************************************************
10 a=RESPR(200)
15 cs=0
20 RESTORE 100
30 FOR i=a TO a+160
40 READ byte:POKE i,byte
45 cs=cs+byte
50 NEXT i
55 IF cs<>12211 THEN PRINT 'error in data':STOP
60 SBYTES mdv1_ldef_bin,a,200
70 CALL a
100 DATA 67,250,0,10,52,120,1,16,78,146
101 DATA 78,117,0,0,0,0,0,1,0,10
102 DATA 4,76,68,69,70,0,0,0,52,120
103 DATA 1,22,78,146,102,42,12,67,0,1
104 DATA 102,34,32,110,0,24,52,118,136,2
105 DATA 213,238,0,32,16,54,160,0,176,54
106 DATA 152,1,103,16,80,136,177,238,0,28
107 DATA 99,230,112,249,78,117,112,241,78,117
108 DATA 18,0,83,64,38,73,82,139,82,74
109 DATA 82,139,24,54,160,0,0,4,0,32
110 DATA 0,54,0,32,184,0,184,54,184,0
111 DATA 102,208,81,200,255,230,12,54,0,4
112 DATA 136,0,103,18,12,54,0,5,136,0
113 DATA 102,196,8,1,0,0,103,2,82,137
114 DATA 211,193,61,182,136,4,152,0,120,3
115 DATA 112,0,45,73,0,88,78,117,255,255
116 DATA 252,38,2,220,0,0,10,48,0,1
```

```
1                       ;*
2                       ;**********************************************************************
3                       ;* Program to add the function LDEF to SuperBASIC which returns the line
4                       ;* number of the given procedure or function
5                       ;*
6                       ;*
7                       ;*
8                       ;**********************************************************************
9                       ;* Constants, variables, vectors etc.
10                      ;**********************************************************************
11                      ;*
12          00000110    BP.INIT   EQU       $110      vector    - add SuperBASIC procedures
13          00000058    BV.RIP    EQU       $58       SBvariable- maths stack start
14          00000116    CA.GTSTR  EQU       $116      vector    - get strings
15          00000018    BV.NTBAS  EQU       $18       SBvariable- name table start
16          00000020    BV.NLBAS  EQU       $20       SBvariable- name list start
17          0000001C    BV.NTP    EQU       $1C       SBvariable- name table end
18                      ;*
19                      ;**********************************************************************
20                      ;* Link in function
21                      ;**********************************************************************
22                      ;*
23 0 00000000 43FA000A             LEA       DEFTAB(PC),A1       definition table
24 0 00000004 3478 0110            MOVE.W    BP.INIT,A2          link it in
25 0 00000008 4E92                 JSR       (A2)
26 0 0000000A 4E75                 RTS                           and return
27 0 0000000C 0000         DEFTAB  DC.W      0                   number of procedures
28 0 0000000E 0000                 DC.W      0                   end of procedures
29 0 00000010 0001                 DC.W      1                   number of functions
30 0 00000012 000A                 DC.W      LDEF-*              relative pointer to routine
31 0 00000014 04                   DC.B      4                   length of function name
32 0 00000015 4C444546             DC.B      'LDEF'              function name
33 0 0000001A 0000                 DC.W      0                   end of functions
34                      ;*
35                      ;**********************************************************************
36                      ;* Get parameters onto the maths stack
37                      ;**********************************************************************
38                      ;*
39                      ;*
40 0 0000001C 3478 0116    LDEF    MOVE.W    CA.GTSTR,A2         get string parameter onto stack
41 0 00000020 4E92                 JSR       (A2)                A1 now equals new start of stack
42 0 00000022 6624                 BNE.S     EXIT                exit if error
43 0 00000024 0C43 0001            CMP.W     #1,D3               1 parameter required
44 0 00000028 6622                 BNE.S     PARMERR             error if not 1
45                      ;*
46                      ;**********************************************************************
47                      ;* search the name table and list for the required name
48                      ;**********************************************************************
49                      ;*
50 0 0000002A 2B6E0018             MOVE.L    BV.NTBAS(A6),A0     start of name table
51 0 0000002E 3478 0002    LOOP    MOVE.W    2(A6,A0.L),A2       offset into name list
52 0 00000032 D5EE0020             ADDA.L    BV.NLBAS(A6),A2     add start of name list
53 0 00000036 1036A000             MOVE.B    0(A6,A2.W),D0       D0=length from name list
54 0 0000003A B03A9801             CMP.B     1(A6,A1.L),D0       is it the same length as string entered
55 0 0000003E 6710                 BEQ.S     CODDBE              if so it could be the one
56                      ;*
57 0 00000040 5088         NEXT    ADDQ.L    #8,A0               next entry in name table
58 0 00000042 B1EE001C             CMPA.L    BV.NTP(A6),A0       check for end of name table
59 0 00000046 63E6                 BLS.S     LOOP                repeat if not the end
60                      ;*
61 0 00000048 70F9         NFOUND  MOVEQ     #-7,D0              'not found error'
62 0 0000004A 4E75                 RTS                           return the error
63 0 0000004C 70F1         PARMERR MOVEQ     #-15,D0              'bad parameter error'
64 0 0000004E 4E75         EXIT    RTS                           return the error
65                      ;*
66                      ;* if reached here entry is the same length
67                      ;*
68 0 00000050 1200         CODDBE  MOVE.B    D0,D1               save length
69 0 00000052 5340                 SUBQ.W    #1,D0               reduce length by 1 for loop
70 0 00000054 2649                 MOVE.L    A1,A3               (A3)=length on stack
71 0 00000056 528B                 ADDQ.L    #1,A3               (A3)=1st ASCII chr$ on stack
72 0 00000058 524A         CHRCHK  ADDQ.W    #1,A2               next ASCII for name in list
73 0 0000005A 528B                 ADDQ.L    #1,A3               next ASCII for name on maths stack
74 0 0000005C 1836A000             MOVE.B    0(A6,A2.W),D4       D4=ASCII chr$ from name list
75 0 00000060 00040020             OR1.B     #$20,D4             make it lower case
76 0 00000064 0036 0020 3B00       OR1.B     #$20,0(A6,A3.L)     make chr$ on stack lower case
77 0 0000006A B83B 8800            CMP.B     0(A6,A3.L),D4       is the chr$ the same
78 0 0000006E 6600                 BNE.S     NEXT                if not same try next name in list
79 0 00000070 51CBFFE6             DBF       D0,CHRCHK           if same check other chr$
80                      ;*
81                      ;**********************************************************************
82                      ;* if reached here must have found required name
83                      ;**********************************************************************
84                      ;*
85 0 00000074 0C36 0004 8800       CMPI.B    #4,0(A6,A0.L)       is it a SuperBASIC procedure
86 0 0000007A 6712                 BEQ.S     FOUND               yes?
87 0 0000007C 0C36 0005 8800       CMPI.B    #5,0(A6,A0.L)       is it a SuperBASIC function
88 0 00000082 66C4                 BNE.S     NFOUND              error if neither
89 0 00000084 0801 0000            BTST      #0,D1               check bit zero of length
90 0 00000088 6702                 BEQ.S     EVEN                even if zero
91 0 0000008A 5289                 ADDQ.L    #1,A1               restore stack: 1 if odd length
92 0 0000008C D3C1         EVEN    ADD.L     D1,A1                              +length
93 0 0000008E 3DB6 8804 9800 FOUND MOVE.W    4(A6,A0.L),0(A6,A1.L) return
                                                                  SuperBASIC line number on stack
94 0 00000094 7003                 MOVEQ     #3,D4               return word
95 0 00000096 7000                 MOVEQ     #0,D0               no errors
96 0 00000098 2D49 0058            MOVE.L    A1,BV.RIP(A6)       restore start of stack
97 0 0000009C 4E75                 RTS                           and end
98                      ;*
99                      END

****** TOTAL ERRORS    0 (line    0)
**** TOTAL WARNINGS    0 (line    0)
     memory usage      8 kbytes
```

```
BP.INIT         00000110  12
BV.NLBAS        00000020  16
BV.NTBAS        00000018  15
BV.NTP          0000001C  17
BV.RIP          00000058  13
CA.GTSTR        00000116  14
CHRCHK          0 00000058  72
CODDBE          0 00000050  68
DEFTAB          0 0000000C  27
EVEN            0 0000008C  92
EXIT            0 0000004E  64
FOUND           0 0000008E  93
LDEF            0 0000001C  40
LOOP            0 0000002E  51
NEXT            0 00000040  57
NFOUND          0 00000048  61
PARMERR         0 0000004C  63
```

passed parameter, after making both lower case. If the Ascii is not the same then control jumps back to the search through the Name List.

If the required name has been found (ie all Ascii characters match) then the type of the name is checked to see if it is a SuperBasic procedure or function. If it is neither of these then the error 'not found' is given. If the name is found then the stack is adjusted for odd or even length paramter names and the start line number of the procedure or function is put onto the stack. This is then returned as a word to SuperBasic.

SOFTWARE FILE

MEDIA MANAGER SPECIAL

INFORMATION:
Program: Media Manager Special Edition **Price:** £49.95
Supplier: Digital Precision Ltd., 222 The Avenue, Chingford, London E4 9SE. Tel: 01-527 5493.

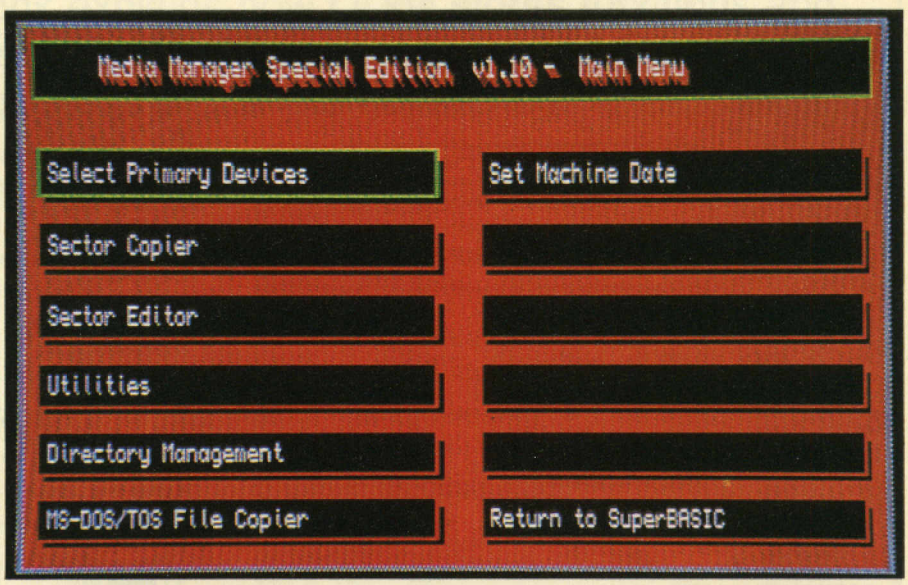


Figure one: The main menu.

There was an earlier version of *Media Manager*, about which the less said the better. It performed certain functions adequately, and gave many of its users a hard time. When you have lost a file and the recovery program gets it back for you, you are likely to feel magnanimous. But there was an obvious need for an upgrade to this particular program. "Upgrade" tends to imply a minor rewrite of parts of a program, and the addition of a few features, but that proved to be unworkable here. It was a case of "back to the drawing board", with a fresh sheet of paper and a revised specification. It would not be possible to cover the program features adequately in one issue of the magazine, so this first part deals mainly with the general layout of the program, and the recovery of corrupted files; the second part will deal with the file-transfer utility operations.

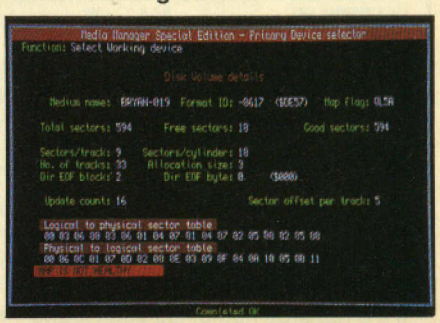
Media Manager Special Edition is the result of collaboration between two of the QL world's most notable personalities — author Chas Dillon and supplier Freddy Vachha. Neither can be easily categorised. To anyone acquainted with them both, the immediate thought would be that the program was not developed in the quiet of a library reference room. There was much discussion, not all of a peaceful nature, and the result is a product which has a strong air of having been well thought-out. The interface with the user alone is something one could spend time playing with, and the operational modules behind the outer face are a well-integrated collection.

If you are familiar with the IBM PC, you are likely to come across a suite of programs called *Norton Utilities*, which were generated by another colourful personality, Peter Norton — who is noted for parking his photograph prominently on his product packages. Media Manager aims to perform a similar set of functions for the QL. The requirements are the same in many respects, but there are significant differences in the way drives are handled on the QL which make the work more difficult. By repute, the QDOS operating system owes a lot to MS-DOS, although this might not be obvious. There is an impression that the firmware driver for the microdrives was written when MS-DOS was around, but the disc driver was an attempt to get away from this influence. Whatever the reasoning behind the drivers, they differ sufficiently to be considered as very different propositions when writing file-recovery routines.

The initial need for a program like Media Manager is likely to be the recovery of a corrupted file or volume (which is the general term for a whole cartridge or disc). The program is also usable in a "healthy" situation, though. In particular, the MS-DOS-to-QL (and vice-versa) utility *XOver* is included and is accessed as one of the Media Manager menu choices. As is typical of Digital Precision's products, the software is accompanied by a comprehensive instruction manual; this gives thorough coverage of the various functions, and the usual helpful explanations on the structure of the QDOS disc and cartridge formats. Breathe a sigh of relief, for the manual is "only" 37 pages long, which is appreciably smaller than the less clear manual supplied with the original Media Manager.

> Bryan Davies finds that the new Media Manager does things that the old one couldn't manage.

Figure two: Primary device selector, select working device.



The space taken by the program has also been reduced, a minimum of 384KB being required to run it. It is available on cartridge or disc (both sizes), and is usable on all Thor models as well as QLs, as is typical for DP programs, SuperBasic extensions are loaded first to provide MMSE with necessary additional commands. Users who already have the Xtras file loaded as a matter of course do not need to change, so long as the version of this file being used is 3.10 or later. MMSE will multi-task with other programs. DP emphasise the disc uses for the program, because it is generally harder to recover information from discs than from cartridges. Whatever the storage medium, it is a golden rule not to attempt recovery on the corrupted volume itself, as this is a sure way of increasing the size of the problem. The first act should always be to copy the information from the bad volume onto a good one, so that the recovery work can be done on the latter. As you might expect, the wisdom of doing this was proven during the course of the review, when the working copy disc of MMSE became corrupted.

You can't go through life as a computer user without occasionally meeting "bad or changed medium", "not found", or "read/write failed". With luck, re-inserting cartridge or disc in drive will cause the message to go away. If not, you have to consider whether a) the problem is mechanical in nature and affecting all columes, and b) it is specific to the particular volume and likely to be caused by data on it being corrupted. With microdrives, mechanical derangement should be top of the list of possibilities. You can be virtually certain that the rubber rollers on the drive shafts will wear, get covered in tape debris, and ride up the shafts, given sufficient use and time. The read/write heads also collect dirt and become worn, but they are unlikely to trouble most users. If you don't service the drives periodically, you can expect the messages. Disc drives are much less prone to becoming unserviceable, but failure to use a head cleaner at regular intervals is a sure way of inviting trouble in the long run.

A careful user can avoid mechanical problems, and can also steer clear of database troubles, by making sure to close files before moving on to another job or switching off. To a large extent, one can also avoid problems which are initiated electrically; effective suppression of the mains supply will considerably reduce the likelihood of lockups. There remains the corruption problems which are largely outside the control of the user, such as crashes caused by software bugs or sudden loss of the public electricity supply.

Individual files on a volume are of little use in themselves, because access to them is via a reference area which is present on *all* formatted volumes, and problems with this area can render some or all files unavailable for use. The terms "map" and "directory" will be met in connection with

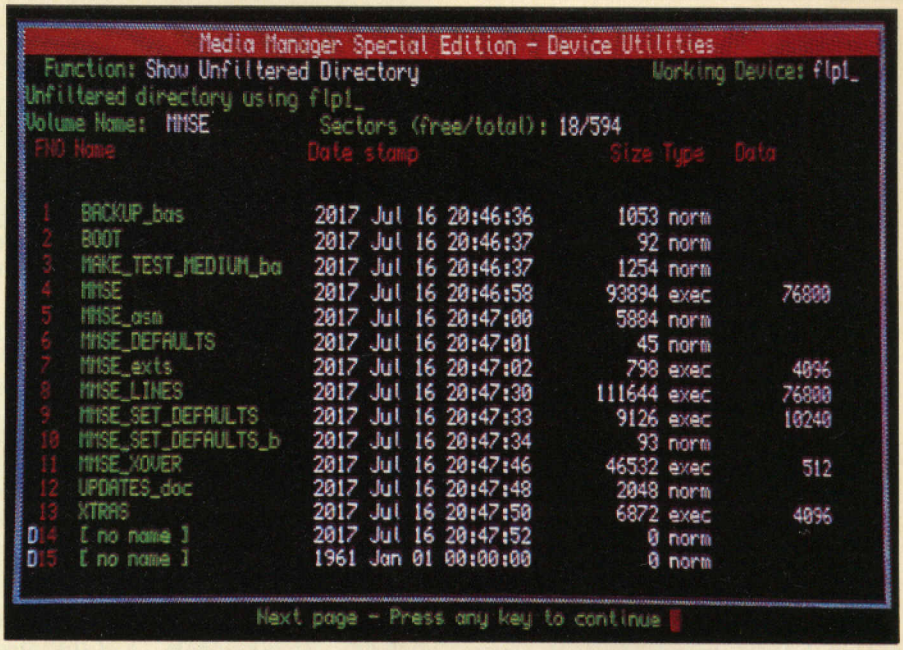


Figure three: Device utilities, show unfiltered directory.

QL cartridges and discs. Briefly, the map contains a record of what is where on the volume, and the directory lists the files contained on the volume. You don't have to erase a file from a volume to get rid of it; all that is necessary is to delete the reference to it in the directory.

What is erased differs between cartridge and disc; it is more difficult to find deleted files on disc than on cartridge. As far as application programs are concerned, any file which does not appear in the directory does not exist.

Deleting

The act of erasing (deleting) a file leaves the actual file intact, but it passes the message to the operating system that the space taken by the file is now available for use by other files. This means that, until other data is written onto the area occupied by the deleted file, there is a good possibility that the file can be retrieved. All that is required is to rebuild the directory entry for the file, in the correct position in the directory, and to identify its physical location on the map. Once one file has been overwritten by another, recovery becomes either difficult or impossible. For this reason, the best thing to do, as soon as trouble is detected, is to stop using the suspect volume, before anything else can be written to it.

The list of functions on the first page of the instruction manual seems a good place to start looking at the program. What is offered is the ability to obtain intelligent directory listings; perform selective file operations; restore deleted or corrupt files; sort directories by name, date or size, in ascending or descending order; read from and write to, alien discs (in MS-DOS/TOS formats); perform direct sector reading, editing, and writing operations on microdrives, or any disc (alien or QL format), and obtain full diagnostic printouts of media storage maps, for enhanced security.

Two features which become more appreciated as the program gets more use are the context-sensitive help lines and the selection of items on the menu by pressing single keys. The help is not verbose — it is one line at the bottom of the screen — but it is well thought-out, and makes the point without getting in the way and without having to be called-up separately. The single-key menu selection does not apply to the main menu, but comes into force once the user has got down to the operations of examining or working upon files and volumes. The manual points out that a rhythm of keypresses can be developed for long recovery jobs, with the presses becoming more or less a reflex action, with no real need to look at the menus.

A file of parameter defaults can be loaded automatically when the program loads. You can change the defaults from the main menu, once the program is running, but free-standing routines are provided to enable the default file to be customised to the user's satisfaction. Data can be printed to a printer, or to a file. The output device for reconstructed files can be mdv, flp or win (not a printer). A SuperBasic program is provided for the purpose of writing a set of sample files to disc or cartridge to experiment with.

The source code of the microdrive access routines is supplied, for those users who are in really deep. The companion to this file is one of extensions, which is created by passing the first file through the *HiSoft* assembler. Initial releases of MMSE will have a line-numbered version of the program file, a somewhat unusual step to cater for the possibility (remote, according to DP) that the normal program

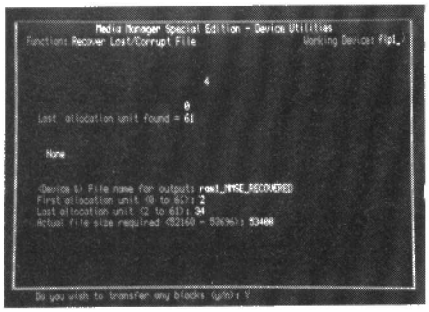


Figure four: Device utilities, recover lost/corrupt file.

file crashes; should it do so, you are asked to run the line-numbered version and try to repeat the performance, noting the line at which the crash occurs.

A backup routine and the usual Quill file of updates (with nothing to say at this time) completes the set. The backup routine needs to be used, because it puts a zero-length-name file onto the working volume, and this file is looked for by the MMSE program. This does not prevent a working copy being made using the Copy or WCopy commands, but instructions are given for creating the zero-length-name file, since the copy commands won't do this. Provision has been made for the addition of other function modules, without MMSE itself having to be changed. Once MMSE is running with the main menu displayed, the program disc can be removed; it will only be needed again if the File Copier option (or other options released in future) are required.

From the start, availability of functions is indicated by colour — when text is white, the option is available, but items with green text are not available at the current point in the procedure. Options are contained within two columns, each of six blocks, and the "cursor" is a green border to fit around the blocks (**Figure 1**). There is no need to use the up/down cursor keys to move the cursor, since the left/right keys step the cursor from side to side between the columns, highlighting all options in sequence. Although no numbering is displayed on the options, the cursor can also be moved to any of the upper ten options by pressing the appropriate numeric key, 0-9. Selection is by means of the space bar or ENTER. A line at the bottom of the screen gives brief information on the option currently outlined by the cursor. A welcome feature, keeping operaton in-line with current fashion on various computers, is that the ESC key can usually be used to step back to the previous menu, escaping from the current one; this is not possible once certain operations have been initiated. Where you are requested to answer a question with (Y/N), pressing the ENTER key is equivalent to pressing the first choice key, which gives consistency with programs such as *The Editor*. Another consistent feature is the use of F4 to refresh the screen.

The cursor is automatically placed over the option which is deemed to be the one most likely to be needed, so you don't have to waste time driving the cursor around every time you select an option. Care has to be taken to provide a normal QL cursor on each screen, making the use of CTRL+C to switch to SuperBasic or other programs possible at all times unless some critical operation like formatting a cartridge is in progress. The initial options available are limited to selecting primary devices, setting the QL date, using the file copier routine, and returning to SuperBasic. The file copier will be dealt with in part two of this review. You cannot get on with the job in hand until the primary devices have been selected — working, output and print.

You are posed a question, after selection of the working device, and it deserves some thought. Do you wish to write to the selected device, or not? As noted before, it is recommended that a faulty volume is not disturbed in any way. The volume may not be faulty, or have only minor errors on it, and you intend to make some changes to it, in which case you would opt to be able to write to it. Cartridge and disc are treated differently; the dodgy nature of cartridges dictates that a copy be made of the "target" one (in the working drive straight away, and this is done to ram, with brief details of the cartridge being displayed on-screen. You are warned that this process — which nominally takes a few seconds — can be lengthy, if the cartridge is in bad shape. With a disc, a complete ram image is not necessary, and you are first presented with an analysis of the basic parameters — tracks, sides and sectors. You are free to change tracks or sides at this point. More comprehensive information is then supplied, such as the total number of sectors and the number of them which are good, the update count (how many changes have been made to the disc since the last format), and sector tables. At this stage, a message containing the words NOT HEALTHY is forewarning of work to come (**Figure 2**). Note the MMSE is looking for corruption in map and directory areas; if you cared to copy perfectly good map and directory information from one volume to another, there would be no reason for the program to say something is wrong.

Endless description of menus will not make the uses of any program too clear to readers, so what follows is an account of the recovery of material on a particular volume. A disc is used for the example, but the procedure is basically the same for a cartridge. The particular disc had been corrupted to the point where a normal directory request produced the sector count, and nothing else, apart from the drive indicator being continuously alight. The disc had been formatted in the formal fashion, then loaded with files using the file-generation routine supplied with MMSE.

The disc map had then been replaced by another one, that did not pertain to the files on the disc; various files had been deleted, and others copied onto the disc. Something was obviously wrong, since no files were shown as existing when the DIR command was issued. When the primary device selection was made, the Disk

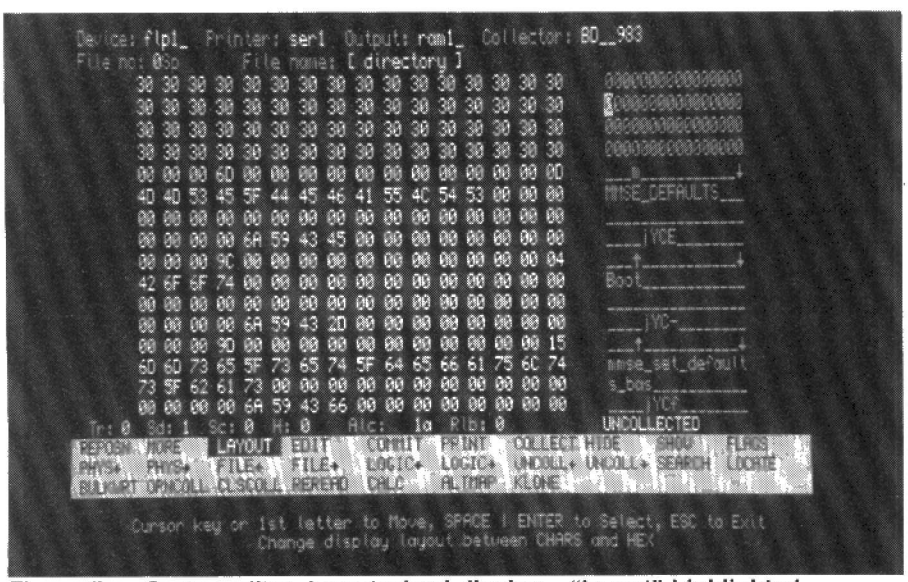


Figure five: Sector editor, hexadecimal display – "layout" highlighted.

Volume information displayed was plausible, and there was no NOT HEALTHY message. The next step was to return to the main menu and select Directory Management, followed by Show Directory. This is where a problem became obvious; three files were shown as being on the disc, but they were beyond the "Logical End of Directory". That is, there was nothing at all in the directory, meaning no files on the disc, yet three files were said to be on it. Not only that, but the files were numbered 17-18-19, indicating that there were, or had been, other files there.

The directory that had been obtained was of the "filtered" type, which is like the SB DIR command and does not show deleted files and blank entries. A more

relevant listing was obtained via the Utilities and Show Unfiltered Directory options (the example in **Figure 3** is not for the disc referred to here). This showed 15 files before the Logical End of Directory, number 16 immediately after the end, the three already found (17-19), and four more (20-23). 16 and 20-23 had "no name" as the file names; this indicates the length of the file name is illegal. The files actually within the directory area had file lengths which were impossibly long, and names which obviously had not been given to them by me (most of them were the same, anyway).

By now, it was clear that there were quite a few files, and that the directory was a complete mess. As a precaution, the Sector Copier option was chosen from the main menu; this automatically makes a bit-for-bit copy of the original volume onto another one, so that recovery work can be done on the copy. Since the copy is exact, down to the volume name and format details (this disc had been formatted to 36/612 sectors), it can be substituted for the original in the working drive, without MMSE getting upset.

Several recovery procedures are offered, to cope with the severity of the problem. Using the Recover Lost/Corrupt File option (**Figure 4**) from the Utilities menu is first choice, but may not be effective, if the information you have on the lost files is insufficient. For instance, you need to know the File Number; to avoid possible unwanted material in a recovered file, it is desirable to know the length of the file to be recovered. If you do have the information, you just tell the program to "go to it", and wait for your file to be churned out. Otherwise, you need to go to the next, less-automated level of options.

We have now worked our way to the "low level" Sector Editor function, having found that the relatively automated operation of the functions prior to this may not be fully effective when the map and directory areas are bad in some way, and knowledge of the lost files is insufficient. There are a considerable number of options on the menu now (**Figure 5**), and they can be chosen by the cursor keys, or by pressing the first letter of the option — if several options start with the same letter, you keep pressing the letter until alphabetic progression gets you to the one you want. This is where convenience of selection is needed, because most time is going to be spent here when a volume is badly corrupted. The manual advises the user to develop a rhythm of keypresses, particularly once the collection of file sectors has been started.

For most users, recovery will be easier for a text file than for a program one, because they will be able to recognise some of the file contents. Choosing Utilities and Show Allocation Summary allows you to see what each Allocation Block contains. This will, for instance, let you know which blocks contain the beginnings

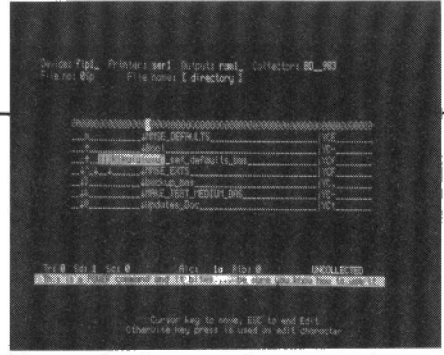


Figure six: Sector editor, commit message: This is a POWER command.

of files, so that you know where to start when piecing files back together again. If you know a file name, or some text at the start of the file, you can use the Sector Editor Locate or Search functions to find files. Locate looks only in the file header, so it is the function to use when looking for a file name, whereas Search looks all through, and is therefore more comprehensive, but slower.

The files created for you by the example routine have nice simple names, starting with "FILE:A". Repositioning to the start of the disc (Track 0/side 0/sector 0) and using Locate soon found the start of FILE:A, even though the directory showed the name as something quite different. Consider the possibility that more than one file has ended up with the same name; this can be checked by using Locate again, from the point where the first file-start has been found. At this point, the thing to do is Open Collector — open a file (in ram for greatest speed of use) in which to collect the pieces of FILE:A. Not surprisingly, DP recommend that you use The Editor for "massaging" recovered files; having used the program for a couple of years, I would not argue with the suggestion, as it seems the most suitable tool for the job. Bearing in mind that MMSE and The Editor can be multi-tasked, progress with recovery can be monitored by switching to The Editor and loading recovered files, to see that they look as they should; gaps can be filled, corrections made, and unwanted material (likely at file ends) can be deleted.

Collection can be done piece-by-piece, using Collect, or in larger chunks, using Bulkwrt (bulk write). The latter automatically uses the Locate function to find the start of a file, provided you have the name to give it, and the name has not been corrupted on the disc. You are asked for the number of sectors to be collected, so that it can calculate how many collection operations to do. You may have no idea of the file size, but it is worth trying to recover a file with Bulkwrt, to see whether or not it looks like being effective. When a file is obviously in such a mess that only visual linking of its contents makes sense, you use the Collect option instead.

The start of the file is found by the Locate function, if possible; if not, the first known piece of text can be found by Search. When the latter approach has to be taken, the Logical move options can be used to move backwards and forwards within the logical file structure (as opposed to moving physically step-by-step through the disc, which is unlikely to relate to where many of the pieces of a file are recorded). In the example disc being recovered, the file names still existed; once the first one had been Locate-d, a file was opened by the Opncoll (open collector) option, and the first Collect operation initiated. The Allocation Blocks are divided into a-b-c sections, and you are assured that data will always be found in abc order, except in circumstances (which usually won't occur) where disc sectors have been corrupted one at a time.

Completion of the first Collect operation leaves you with a screen displaying the next Logical point in the file, as the program sees it. If it looks the right place to you also, you simply press the ENTER key three times and this piece is collected automatically too, and so on until you reach a display of data which is obviously at or beyond the end of the file. The collector file is then closed with the Clscoll option. It's worth having a look at the file in The Editor, or by Copy-ing it to the screen, to satisfy yourself that it is basically correct and complete.

All the remaining files from the example disc were recreated this way. Some of them required attention in The Editor, mainly to remove unwanted portions from the ends. Once having got into the swing of it, the procedure was straightforward and reasonably speedy. When the volume has been badly messed about, recovery of any sizeable amount of information is not going to be a five-minute job, but the time taken on the sample seemed in proportion to the size of the task. The disc used for the copy of the original, bad disc was not formatted beforehand, and had a number of other files on it.

Nothing had been done to remove these files, so they appeared on-screen when the Show Allocation Summary option was used. The sample disc had been formatted to 594 sectors, against the 1440 of the copy disc, and "other" files were clearly in the area above the 594 sectors; in this instance, these extra files were ignored, and they didn't upset recovery of the sample files, but it would have been possible to recover those portions of these extra files that had not been overwritten when the copy was made. For the brave, there is the facility to correct Directory entries; as illustrated by the message in **Figure 6**, "This operation is a POWER command and it bites!", so should be treated with caution.

A further test was made with two existing document files, one from The Editor and one from *text*[87]. Parts of the files were overwritten, and the directory entries for them were corrupted. Three different approaches to recovery were taken, and they produced near-identical results. The recovered files were loaded into the programs they were written in and they looked just as they ought to.

The remainder of the program is dealt with in Part 2 of this review.

ARCHIVE POWER

Robin Stevenson starts a four-part series in which he explores the strengths and weaknesses of Archive.

QL owners have had access to *Archive* for as long as they have had QLs. All must have tried a few things with GAZET.DBF, and many will have developed applications of their own. However, my impression is that Archive is rarely thought of as a real computer language. Its full possibilities are not even exploited in database applications — its 'natural home'. In other computing areas it tends to be ignored completely, even though it has much to offer. This short series is intended to demonstrate some of the programming possibilities of Archive, utilising its strengths, and providing solutions to some of its weaknesses.

What are the strengths of Archive? Why use it instead of SuperBASIC or C? The primary justification must be speed — not speed of execution, certainly; it is well down those particular stakes, but more important in many instances is speed of development. The reason Archive may be well up here is because it is a 4GL — a Fourth Generation Language. In other words, an extra layer of the programmer's job has been done already. Time consuming problems have already been solved, and the solutions made available in an easy-to-use form.

Psion saw the storing of information on disk and its subsequent manipulation as a prime problem, and set out to solve it. The ability to search, order and select records, use

ARCHIVE AND OPL

Development of the *Archive* language, along with the rest of the Psion programs, more or less halted over three years ago. At that point the company's energies began to be taken up instead by the Psion Organiser, in its various configurations. A vital part of the Organiser — built into its operating system in the way *that SuperBasic is in the QL — is OPL*, the Organiser Programming Language. However the Psion software team did not start from scratch with OPL; rather they started from the Archive language. The result, while not identical to Archive, is very much a dialect of the same language.

With the launch of Psion's Mobile Computers OPL has been given yet another platform, built into the operating systems of the MC200 and MC400s, with their exciting multi-tasking, windowing, graphics user interface. If these machines take off in the way Psion hopes, OPL looks set to expand in significance over the next few years.

So in what ways is OPL similar to Archive? Like Archive it is based entirely on procedures. Layout is the same, ends of statements being denoted by colons or ends-of-lines. Mathematical operators are identical. The keyword list is broadly similar, especially input and output, as is the approach to data files.

The differences are not slight, but are mostly differences of detail. A number of keywords have been shortened, and there are a number of new ones, both procedures and functions, working at low level (eg peek and poke commands, free device space, etc.) and at high level (eg trigonometry/log functions and built in menu function). An integer variable type is available. Variable assignments don't need a LET keyword. Functions returning a string end in a dollar sign.

Other differences will have more effect on programming style. Procedures are all stored individually on the storage device, so each can be easily shared between many applications. They can return a value, as in SuperBasic FuNctions. Procedures are stored in ascii form, but also as 'translated', executable files. All variables must be declared as either LOCAL or GLOBAL. The latter are available to procedures called by the 'parent', but like LOCAL variables, cease to exist when the parent procedure is completed.

There are a number of useful additions to the control structures. A new structure, DO..UNTIL<exp> is available, similar to WHILE<exp>..ENDWH, but testing for an exit after the body of the loop. Both loops can be exited using the BREAK command. IF..ELSE..ENDIF is also enhanced by addition of ELSEIFs, allowing easy testing of a list of options. You can also GOTO a labelled line.

The main ommission on the Organiser is the lack of the more powerful file commands. There are no indexing, selecting, or locating options, which must limit its usefulness as a database language. Less surprising, given the organiser's screen, is the absence of the screen editing and input commands. However there are USR functions, as in *Archdev*, allowing links to other machine code programs.

The MC Mobile Computers contains an enhanced version of OPL, which presumably allows access to their special features — the touch-pad pointing device, resizable windows, multi-tasking, and all the other wonderful things Psion are cramming into their new boxes. If you are planning a mobile addition to your hardware, in the shape of one of Psion's offerings, familiarity with Archive will make learning OPL a doddle.

disks/microdrives as virtual memory, input and modify data, and use several files at once give it the database leaning, although plenty of other applications can make use of those features.

Other aspects that make for easy, quick program development include the programming environment, error trapping, screen 'painting', and the printer driver. Programmers will have their own priorities over these and other features, and all will have a matching list of limitations and grumbles. Nevertheless, they represent facilities that take huge programming effort in a more 'conventional' language. By getting rough prototypes up and running in a few hours, and decent working versions soon after, applications become possible which would be completely unjustifiable otherwise.

Obviously this series cannot deal with your specific problem, so instead I will be developing a selection of more general 'Desktop functions': a notepad cum card-index file; a calculator; a calendar and a to-do list cum appointments diary. These will all be integrated into one Archive application, but of equal benefit will be the toolkit we develop on the way. This will be a range of re-usable procedures for menu and prompt handling; file, window and printer control and coping with dates.

The program we develop will be presented each month as listings. Familiarity with Archive EDIT is essential. Suffice it to say here that each procedure needs to be created via the <F3> command. There are added complications to printing an Archive program: one is the use of 'screens' created via the SEDIT command. As well as improving screen appearance, these provide much better input options than are possible purely through program code. They will be an integral part of our program, and will be printed in the form of anotated screen shots.

Another complication derives from the problem of QL memory. Owners of beefed up, fat cat QLs may have forgotten about this, but an unexpanded QL only has some 20K available after Archive has swallowed up the chunk it uses. Our program will be usable on an unexpanded QL, but only just, and only through various devious means. One of the techniques used is to 'chain' the programs. Unlike SuperBASIC, an Archive program can run another Archive program — without losing the existing memory or file variables. We shall end up with three linked program files. One is a short initialisation program. Secondly, the NOTEPAD program will also provide the main menu. Finally the CALENDAR and TO-DO LIST programs are combined. The CALCULATOR is to be available from any standard menu, so it is present in both main programs.

For unexpanded QLs memory management becomes significant, especially as Archive doesn't manage very well. There is a gentle decline in available memory as programs are run, ultimately causing 'out of memory' errors. There are ways of optimising memory, however. A couple of kilobytes can be gained by clearing out SuperBASIC, in the bootprogram. If you replace the EXEC_W MDV1_ARCHIVE. Alternatively (or in addition) you can sacrifice some of the less-used functions in the program. This will not apply to anything in this month's listings, but you may decide, for example, that you can live without the CALCULATOR option, developed in part three, or some of the more obscure features added to the NOTEPAD next month. One advantage of typing in and understanding a program in this way is that you can reach you own compromises, rather than being stuck with mine.

Before we go into the details of the program we shall discuss a number of design decisions which have been incorporated into the toolkit procedures.

Consistency in screen layout is important. Learning to use a program is in large part learning where on the screen to look to find out what is going on, and what can be done. Under the system provided here, all available actions are shown in the prompt area. Following Psion's lead, we shall use the top few lines of the screen. However, in addition to menu choices, we will also use the prompt area for text responses. There are distinct benefits in keeping all tookit actions within a well defined area of the screen in this way. Application screens can then be designed with that in mind, and there is no danger of toolkit procedures trying to write over data, or vice versa. We shall also use the bottom line for general system information — in this case available memory, date and time.

To summarise, of the 25 lines available on screen, Line 0 is to be used as a heading. Lines 1 and 2 are the normal prompt area. Line 3 is a delimiting line. Lines 4 to 23 are available for application data. And line 24 is for system information. Our program assumes screen MODE 0,8. The main problem with 80 columns of tv is the top and sides drifting out of sight. To counteract this, minimum use has been made of these areas.

The other convention adopted relates to writing safe, re-usable toolkit procedures that will not affect, or be affected by, other procedures used alongside them in an unexpected way. As Archive procedures cannot return a value (as in SuperBASIC FuNctions) the only way information can be passed back is by global variables. To keep an orderly, predictable pattern to this we shall restrict ourselves to a numeric and a string variable, respectively called ANSWER and ANSWER$, and the numeric variable YES for when a true or false value is required.

To minimise the 'side effects' each procedure must exist as much in isolation as possible. There are three primary rules for achieving this ideal. 1) All information the procedure needs is passed as a parameter 2) All intermediate variables should be local 3) Communication back to the calling procedure should take place only by one or more of the above global variables. This is not just purist mumbo-jumbo. It is part of making maintainable code. In this way the entire effect of any changes to a procedure are visible in the procedure itself.

This month's listing is the first half of the

Continued on Page 32

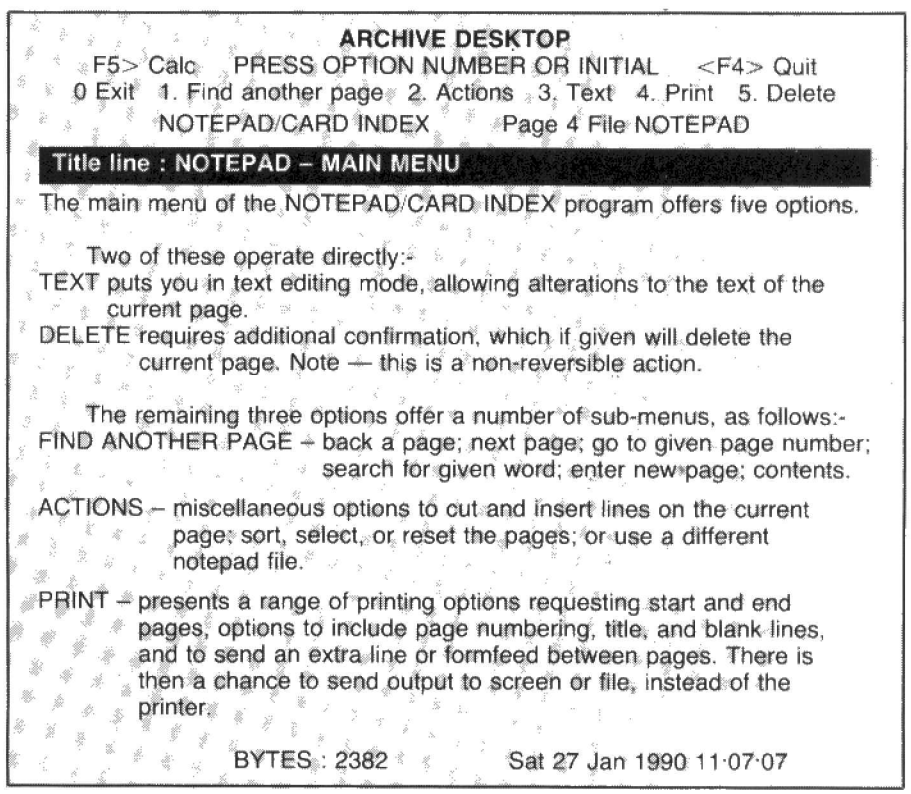


Fig. 1: The main menu screen of the Notepad program.

NOTEPAD program. Once typed in it will be usable for note entries, but very little else. Next month we will add a range of useful features to it, and discuss the various ways it can be used. This month we concentrate on the toolkit procedures.

The listings are printed with all user-defined names in upper case, making for more readable code. The editor automatically translates Archive commands into lower case, so by leaving caps-lock on for all except string literals (text in quote marks), you will also spot mis-typed Archive statements, as they will remain in upper case.

Enter the listings in the order printed (Archive will display them in alphabetical order). Many procedures rely on those developed earlier. As you type in each procedure you can test it and see how it works. This not only makes mistakes easier to detect, but also aids understanding. Do not enter the REMarks unless you really have got time and memory to spare.

The first procedure, START, is a temporary one, for use only until next month. It initialises two global variables (although not with their final values), and attempts to run the NOTEPAD. Obviously this last stage will not work until the NOTEPAD procedure is typed in. Run it at least once at the start of any editing session, to

Continued on Page 35

Listing for Part One - Notepad procedures

```
proc START
  let TODAY$=date(1)+"        "
  let HEADING$=chr(15)+chr(20)+chr(0)+chr(0)+chr(80)+chr(25)
  mode 0,8: sload "NOTEPAD"
  error NOTEPAD
  error SHUT;"": mode 1,8
-------------------------------------------------------------------------
proc WINDOW;WID,DEP,COL,ROW
  print chr(20)+chr(COL)+chr(ROW)+chr(COL+WID)+chr(ROW+DEP);
  endproc
-------------------------------------------------------------------------
proc SHUT;FNAME$
  let CURRENTFILE$=""
  if FNAME$=""
    while 1: close : endwhile            : rem CLOSE ALL OPEN FILES
    else : close FNAME$: endif          : rem OR JUST THE NAMED FILE
  endproc
-------------------------------------------------------------------------
proc CENPRINT;LINE,MESS$
  local COL: let COL=40-(len(MESS$)/2)      : rem FIND START COLUMN
  if COL<1: let COL=1: endif     : rem CHECK IT'S NOT LESS THAN ONE
  print at LINE,0; ink 4;chr(27)+"A"; tab COL;MESS$;
  endproc
-------------------------------------------------------------------------
proc GETCHOICE;MESS$,MAX,INIT$
  local KEY$,ANS$,F$             : rem FIRST PRINT THE STATUS LINES
  let F$="BYTES :"+gen(memory(),7)
  print HEADING$; at 24,10;F$; at 24,42;TODAY$;"  ";time();
  let F$="PRESS OPTION NUMBER OR INITIAL"
  CENPRINT;1,"<F5> Calc   "+F$+"   <F4> Quit"
  CENPRINT;2,MESS$                    : rem THEN DISPLAY THE PROMPTS
  let ANS$=INIT$+"0123456789"(1 to MAX+1)
  while not instr(ANS$,KEY$)                      : rem LOOP FOR INPUT
    let KEY$=upper(inkey())      : rem USE getkey() IF MULTITASKING
    if KEY$=chr(22): let F$=CURRENTFILE$: error CALCULATOR
      let CURRENTFILE$=F$: if F$<>"": use F$: endif
      GETCHOICE;MESS$,MAX,INIT$: return : endif
```

```
    if KEY$=chr(21):CONFIRM;"Quit this prgram"
      if YES: error SHUT;"": mode 1,8: stop : endif
      GETCHOICE;MESS$,MAX,INIT$: return : endif
    print at 24,60;time();
    endwhile                                  : rem END OF INPUT LOOP
  if instr(INIT$,KEY$): let ANSWER=instr(INIT$,KEY$)-1
    else : let ANSWER=val(KEY$)          : rem PUT RESULT IN ANSWER$
    endif
  CENPRINT;1,str(ANSWER,2,0)                  : rem DISPLAY RESPONSE
  endproc
-------------------------------------------------------------------------
proc CONFIRM;MESS$
  GETCHOICE;MESS$+"? (1. Yes  0. No)",1,"NY"
  let YES=ANSWER
  endproc
-------------------------------------------------------------------------
proc ALT;NEWRECORD
  local T$,P: let P=recnum(): sprint          : rem STORE recnum()
  if NEWRECORD: let T$="INSERT"     : rem SET UP APPROPRIATE PROMPT
    else : let T$="ALTER": endif
  CENPRINT;1,T$+" : Press <Tab> or <Enter> for the next line."
  let T$="<SHIFT>+<TAB> back a line. "
  CENPRINT;2,T$+"<F5> completes. <ESC> quits without updating."
  alter : position P             : rem RESTORE POSITION AFTER ALTER
  endproc
-------------------------------------------------------------------------
proc GETSINPUT;DEF$
  let ANSWER$=DEF$
  sinput ANSWER$                  : rem ERROR TRAPPED USE OF SINPUT
  endproc
-------------------------------------------------------------------------
proc GETSTRING;QUEST$,DEFAULT$
  let ANSWER$=DEFAULT$
  WINDOW;80,25,0,0                          : rem SET UP PROMPT QUESTION
  print at 1,0; ink 4; tab 46-len(QUEST$);QUEST$;" : ";
  error GETSINPUT;DEFAULT$                   : rem ATTEMPT TO USE SINPUT
  if errnum() and errnum()<>27              : rem IF IT FAILS, USE INPUT
    print chr(27)+"A";: input ANSWER$
    endif
  endproc
-------------------------------------------------------------------------
proc FOPEN;MESS$,LOG$
  local LOOP: let LOOP=1
  if instr(MESS$,"File Is ")<>1          : rem SEE IF FILE IS GIVEN
    while LOOP                          : rem IF NOT, ENTER INPUT LOOP
      GETSTRING;MESS$+" ( ? for dir )",LOG$
```

```
      if ANSWER$="?"                                  : rem IF DIRECTORY
        GETSTRING;"Directory for",""                  : rem GET DEVICE NAME
        WINDOW;70,17,5,5: dir ANSWER$                 : rem & DISPLAY FILES
        else : let LOOP=0                             : rem OR EXIT THE LOOP
        endif
      endwhile                                        : rem END OF INPUT LOOP
    else : let ANSWER$=MESS$(9 to ): endif : rem OR FIND FILE NAME
  let YES=0: error SHUT;LOG$          : rem ENSURE FILE STARTS CLOSED
  open ANSWER$ logical LOG$                       : rem ATTEMPT TO OPEN IT
  let YES=1: let CURRENTFILE$=LOG$  : rem THESE ARE SET IF SUCCESS
  endproc
-------------------------------------------------------------------
proc NOTEPAD
  local FILE$,OPTION,T$: let REDRAW=1
  let YES=0: while not YES          : rem LOOP TO OBTAIN/CREATE FILE
    error FOPEN;"NOTEPAD -- Required file","NOTEPAD"
    let FILE$=ANSWER$: if errnum()=100       : rem CANNOT OPEN FILE
      CONFIRM;FILE$+" does not exist. Create new file"
      if YES: create ANSWER$ logical "NOTEPAD"
          PAGE:TITLE$:N1$:N2$:N3$:N4$:N5$:N6$:N7$:N8$:N9$
          N10$:N11$:N12$:N13$:N14$:N15$:N16$:N17$:N18$:N19$
          endcreate : append : let PAGE=1: append
        order PAGE;A
        endif : endif : endwhile      : rem END OF OBTAIN FILE LOOP
  last : let PAGETOT=PAGE             : rem STORE HIGHEST PAGE NUMBER
  while 1                    : rem ********** MAIN NOTEPAD MENU LOOP
    if REDRAW:window;80,25,0,0: screen : let REDRAW=0: endif
    let ANSWER$="": print at 3,15; ink 2;"NOTEPAD/CARD INDEX"
    let T$="  File "+upper(FILE$)+chr(126)+chr(126)
    print at 3,40; ink 2;"Page ";PAGE;T$: sprint
    let T$="0. Exit  1. Find another page  2. Actions"
    GETCHOICE;T$+"  3. Text  4. Print  5. Delete",5,"EFATPD"
    let OPTION=ANSWER                    : rem RESPONSES TO SELECTION
    if OPTION=0: error SHUT;"NOTEPAD": return : endif
    if OPTION=1: error PAGEFIND: endif
    if OPTION=2: error PAGEACT: endif
    if OPTION=3: error ALT;0: endif
    if OPTION=4: error PAGETEMP: error PRINTOFF: endif
    if OPTION=5:CONFIRM;"Confirm - delete page "+str(page,2,0)
      if YES: delete : endif : endif
    endwhile                   : rem ********** END OF MAIN MENU LOOP
  endproc
-------------------------------------------------------------------
proc PAGEFIND
  local OPTION,C,P,T$: let P=PAGE: let YES=1
  let T$="0. Exit  1. Back  2. Next  3. Goto  4. Wordsearch"
  GETCHOICE;T$+"  5. Add new page  6. Contents",6,"EBNGWAC"
```

```
let OPTION=ANSWER: if REDRAW: screen : let REDRAW=0: endif
if OPTION=1 and PAGE>1: back : return : endif
if OPTION=2: next     : rem ADD NEW PAGE IF THIS IS THE LAST PAGE
  if eof(): let OPTION=5: endif : endif
if OPTION=3
  GETSTRING;"Go to which page number? ",str(pagetot,2,0)
  locate val(ANSWER$)          : rem CHECK NOT AT 1st, EMPTY PAGE
  if PAGE=0: next : endif : endif
if OPTION=4:GETSTRING;"Search for which word(s)?",""
  find ANSWER$: while found() and YES
    let P=PAGE: sprint :CONFIRM;"Continue searching"
    if YES: continue : endif
    endwhile
  if not found():CENPRINT;1,"NOT FOUND.": locate P: endif
  endif                              : rem GO BACK TO LAST PAGE FOUND
if OPTION=5: first : let PAGE=PAGETOT+1: append
  let PAGETOT=PAGE: screen : print at 3,45; ink 3;"Page ";PAGE
  error ALT;1: endif             : rem CREATE PAGE, & THEN EDIT IT
if OPTION=6: first : next : let C=0: while YES and not eof()
    if C=0:WINDOW;76,19,2,5: cls : endif
    print gen(PAGE,6);" - ";TITLE$: next : let C=C+1
    if C=21: let C=0:CONFIRM;"Continue Listing": endif
    endwhile : locate P: let REDRAW=1:PAGEFIND: endif
endproc
```

ensure the variables are initialised. HEADING$ uses the screen driver to turn off the cursor, and give the full 80 x 25 screen, whenever it is PRINTed. Much use of the screen driver will be made by our program. For a full description see, for example, *Archive Secrets* in QLW October 1989.

And so on to the toolkit procedures: we shall look at their purpose and use and, where applicable, ways they could be extended if so desired. (The parameters required are shown as numeric or string expressions, as in the *User Guide*.)

WINDOW; <n.exp>,
<n.exp>,<n.exp>,<n.exp>

Makes the screen driver windowing facility a little friendlier. The four parameters are the width and height of the window, and its column and line co-ordinates. It could be improved with some range checking, trimming excess lengths and widths to fit the screen.

SHUT; <s.exp>

Is one of the procedures which enable error trapping. It should always be used preceded by the ERROR statement. There are two modes of use: the first is to supply a logical file name as the parameter, which SHUT will attempt to close. If it was not open the error will be caught. The second is to provide an empty filename, which closes all open files. The WHILE loop continues to close files until all are closed. The error resulting from the next CLOSE statement returns control. This is useful, particularly when starting and ending a program, to ensure no stray files are left open.

The global variable CURRENTFILE$ is also reset to empty. This is used to keep track of the logical file name of the file currently in use, enabling 'on-line' procedures, such as the calculator, to restore the file afterwards. Ideally, all the file commands should be both error trapped, and used to update CURRENTFILE$ in this way. However the desktop program almost entirely uses single files, so only closing and opening files are provided for in our toolkit. If required, similar procedures could be provided for USE, LOOK, and IMPORT.

CENPRINT;<n.exp>,<s.exp>

Is another quick and easy procedure. It prints a message in the centre of a line, requiring the line number, and the message as parameters. The start point for the message is calculated from its length. The line is cleared, ink set to green, and the message printed. It is mainly for use by other toolkit procedures, and helps provide neat and tidy prompts.

GETCHOICE;<s.exp>,<n.exp>,
<s.exp> : result in ANSWER

Is one of the most important toolkit procedures, as it is the main point of interaction between the program and the user. It prints the prompts simply by sending a message (the first parameter) to CENPRINT, along with the 'standard prompt' line. The program then loops around waiting for a valid keyboard response. Valid responses are a number between 0 and the number of options (the second parameter); one of the initial letters of the options (which are passed as a string in the third parameter); or one of the built in functions. While it is looping, an on-screen clock is maintained on the bottom line. This is useful in reassuring the user that the program is (a) still running, and (b) waiting for a response. If, on the other hand, you are multi-tasking other jobs at the same time, this kind of manic looping is a good way of grinding everything else to a standstill. If that is likely, use a true multi-tasking clock, and

replace INKEY() with GETKEY(), giving other tasks full use of the processor until you get round to responding.

In addition to the specified menu, GETCHOICE provides access to built-in functions. The idea is that these can be accessed from any normal menu. Two are to be provided — a rapid exit from the program, and a calculator. After calling one of these functions, GETCHOICE must be able to carry on with the original menu request. We will be looking at the programming issues involved in this when the calculator program is developed, but in the meantime identify the two examples of recursion — direct and indirect — employed.

CONFIRM;<s.exp> : result in YES

This is a customised use of GETCHOICE, for occasions when a YES or NO is required. It saves a bit of code, by adding the extra prompts and parameters, instead of having to add them each time. Mainly, though, it provides clearer source code, with the result put into the variable YES. This makes it very obvious what is being done, when CONFIRM is used.

ALT;<n.exp>

Is primarily provided to error trap the use of the ALTER command. It provides one of the few occasions when the ESC key can be safely used, in this case to abandon an edit. If Archive is not waiting for keyboard input, use of ESC causes the program to terminate, handy for debugging, but not much fun in normal use. Also the V2.0 ALTER bug, which can corrupt records, is trapped by restoring the record position afterwards. In addition ALT provides a set of prompts, suitably modified depending on whether the NEW-RECORD parameter is set or not. The INSERT command is not used by this program, due to the lack of control it gives. Instead, each application is responsible for adding its own new records. For example, with the notepad, the first file is an empty record. By going to this record, and using APPEND a new, blank record is created, which can then have its page number added, and be passed to ALT for editing.

GETSINPUT;<s.exp>, : result in ANSWER$, or errnum()
GETSTRING;<s.exp>,<s.exp>: result in ANSWER$

These are used together, to provide a text response facility (what a Mac user might call a dialogue box). GETSINPUT is only likely to be used by GETSTRING, and simply provides error trapped access to SINPUT, a much more powerful command than INPUT, as it allows editing of existing as well as new text. This means the program can offer a default response (GETSTRING'S second parameter), which can be accepted, replaced or edited, as required — just like in the other Psion programs.

Unfortunately, to use SINPUT, there must be an active screen variable of the required name. In other words, you need to decide, when you SEDIT your screen, that you want to use SINPUT at a given place. If Archive can't find such a variable, SINPUT generates an error. Several linked strategies are suggested here to overcome this limitation. First, SINPUT is to use the standard global variable ANSWER$ to make its request. Secondly, any full screen design should incorporate the ANSWER$ variable, starting at postion 1,50 in the prompt area. GETSTRING can then fit the prompt (the first parameter) on the left hand side, to match.

Whenever possible, application programs should keep this screen active, in case it is needed. Finally, as a fall-back position, GETSTRING can check that GETSINPUT ran without error, and if necessary request normal INPUT instead, which provides an answer, but without a default option.

FOPEN;<s.exp>,<s.exp> : result in ANSWER$, CURRENTFILE$ and YES or errnum()

The last of this month's toolkit procedures provides a number of useful facilities related to the opening of a file. The second parameter is the logical file name which will be given to any successfully opened file. It is also the default name offered if the user is to be consulted on which file to open. Under this scenario, the first parameter provides a prompt message, and the user can accept the default offered, enter another file name, or request to see a directory, by entering a question mark. In this case, the device is specified, the directory shown in the data area, and the request made again, until a file name is given.

The other mode of use is to exclude the user altogether. In this case, the first prompt should begin with the words "File Is", followed by the desired physical filename, again with the logical filename as the second parameter. This simply bypasses the user request section. In both cases, the physical filename is put into ANSWER$, and the logical filename into CURRENT-FILES$. If Archive cannot open the file, eg because it doesn't exist, an error will be generated, which should be trapped, and YES will be false. It is up to the calling procedure how to handle a failed attempt.

The remainder of **Listing Two** is the start of the notepad program. Before typing this in you should produce the

ARCHIVE — QL vs PC

Since its inception on the QL, the Psion quartet has appeared on a number of other computers. In its more integrated form as *Xchange*, it is available for PC-compatibles, ICL OPDs, and QL derivatives, among others. For more limited budgets there is the 'not-quite-integrated' *PC-Four* package, for PC compatibles. All of these contain the *Archive* program, more or less as found on your QL. More recently Psion's interest has been elsewhere, and their software has not kept up with the PC market, but if you have to use a PC as well as a QL, the PC-Four or Xchange programs may give you an efficient way of using your existing knowledge to best effect.

In producing the PC version of Archive, Psion decided to sacrifice direct compatibility for the sake of progress. A number of genuine enhancements were made, and also several curious omissions, notably the POSITION, RECNUM() and MEMORY() commands. For this reason you will not easily be able to run a QL Archive program directly on a PC, and the other way round is almost certain to fail without significant modification.

So what are the improvements made on PC Archive? The first thing you will notice is the extraordinary increase in speed. Considering the QL and PC-XT run at similar clock speeds, the difference is surprising, and strongly suggest that Psion failed to optimise the QL code. If you have a fast PC-AT Archive hurtles along, giving much greater freedom and scope for ambitious programming.

In the language itself the biggest change is in holding the database indices in files of their own. This allows up to four different orderings, plus one selection, to be held simultaneously with instant switching between them. Another innovation is 'identifier indirection'. An example will best explain this: the statements LET X$='Y' : LET @X$=25 will assign 25 to the variable Y - being redirected via X$ by the @ symbol. This allows much more flexibility in designing reusable procedures. On the Xchange version there is also record locking and other commands to facilitate use on a Network.

Other than these differences, the two language implementations, like the packages as a whole, are very much the same. PC-Four provides a familiar environment for QL users, and PC Archive offers and instantly available way for a QL Archivist to program a PC compatible.

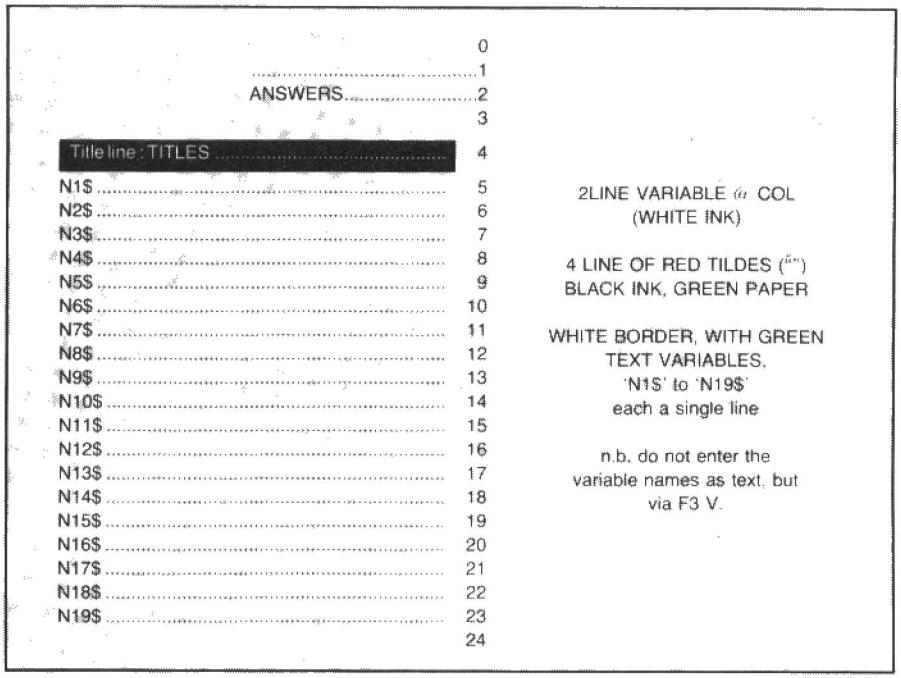


Fig. 2: The Sedit screen needed to start Notebook.

SEDIT screen shown in **figure two**. The annotations give additional guidance on what is intended for this screen. The vital point is to identify which parts are to be typed in directly, and which are to be entered as screen variables. This should present no problem to anyone familiar with SEDIT. On the other hand, many users will not be aware that SEDIT can generate a screen for the full 80 x 25 area. Normally, even in MODE 0 the bottom two lines are reserved by the SEDIT program. However, if you resize the screen to its full proportions before calling SEDIT, the entire screen is available for use. This can be done, with the toolkit procedures loaded, by typing MODE1,8: WINDOW;80,25,0,0: SEDIT from the archive prompt. The problem now is that, within SEDIT, you are writing in parts of the screen that Archive is also using for its prompts. It is not a problem for Archive, as the writing which appears to overwrite your screen at the top and bottom will not appear on the finished screen. It is only a problem for you, remembering what you have put underneath.

Finally you can type in the first two procedures of the NOTEPAD application. Explanation of how to get the most from the notepad and its card index abilities, will be given next month. The NOTEPAD procedure is in two parts. First it attempts to open a file. If it cannot find the file named (as will happen the first time you run the program), it will offer to create a suitable file of that name for you. In this way you can create as many different notepad files as you require. The second half of this procdure is the main program loop, controlling the display, and menu options. The PAGEFIND procedure facilitates finding existing page entries and adding new ones.

The menu will also show options for printing and 'actions'. Do not be tempted to select these yet, as an error will result. Next month we will be adding not just a flexible printing procedure, but facilities to sort and select pages; cut, insert and order lines; and more toolkit procedures.

SOFTWARE FILE

MS-QLINK

Bryan Davies reviews a moderately priced, effective file transfer program from Portugal, which has taken several years to develop.

We do not often review programs written outside the U.K., because we do not have many programs submitted from other countries. There are active users in several countries, perhaps the most well-known to us being those in Germany. Portugal is not the first country to spring to mind as a home of QL programmers but we receive fairly regular letters from a small band of QL devotees there.

One of our correspondents there is Tiago Frietas Leal and he uses both QL and PC. Presumably for his own use initially, he wrote a file transfer program for those computers. He has also written a copy utility, *Discopy*, but that is commented on only briefly here because Leal indicated that the program was not complete at the time he sent a copy towards the end of 1989. It is assumed that it will have been completed by now.

There are already file transfer utilities for PC-QL working, the obvious ones being *DiscOver/Multi/DiscOver* and *XOver*. *MS.QLink* is appreciably cheaper than both; the DiscOver variants cost £29.50 and £39 respectively and XOver is only part of the Solution or Conqueror emulator packages, which cost considerably more. As a sampler, a public domain version of MS-QLink is available for £1 plus a blank disc on which it can be copied. The only functions provided in this version are Directory and View To Screen, for MS-DOS and Qdos.

IMPORTANT
Certain points criticised in this review have been updated promptly by the publisher. A list appeared in QL Scene, May 1990.

INFORMATION

Program: MS-QLink 1.0
Discopy 1.0

Price: MS-QLink £12, MS-QLink PD £1 plus disc, Discopy £7, MS-QLink plus Discopy £17

Supplier: Qfile, Apartado 2110, 1103 Lisboa Codex, Portugal.

Payment by U.K. sterling cheque made out to T.F. Leal. Specify 5.25in or 3.5in disc, single- or double-sided.

Instructions on the full version are in the form of Quill _DOC files, and are extensive – more than 6,000 words. Apart from odd lapses, the author's English is commendably good; the instructions generally can be assimilated rather more easily than some English-written instructions I have encountered. This is not a load-and-go utility; you need to do some reading first. The standard 360KB 5.25in and 720KB 3.5in disc drives are supported and the same operations can be performed on both MS-DOS and Qdos discs.

MS-DOS sub-directories are not supported, but users who do not have hard disc drives will presumably not be unduly interested in sub-directories. The program will not work with QL interfaces which do not permit direct disc sector addressing and the author suggests it may be necessary to upgrade the interface eprom – contact Care Electronics – if you have an early Micro Peripherals or Medic interface. The program can be supplied on 5.25in or 3.5in disc.

MS-QLink uses a run-time Turbo Toolkit to provide the necessary extensions to SuperBasic. It also has a small routine which can be used if your Qdos rom is earlier than JS. The program can be started by either EXEC_W or EXEC and will multi-task with other programs. The command line editor works in the Insert mode for both Qdos and MS-DOS. No doubt that will appeal to many DOS users who have to suffer the deletion of everything they have typed when the cursor is moved left to add or change characters.

Commands and parameters have to be separated by one space, as with MS-DOS. Characters in the code range 32 to 191 are accepted, plus F3/F4/ESC, left/right arrows and CTRL+left/right arrows; the extra keys are for command line editing. The commands have been reduced to one letter for MS-DOS; for Qdos the same letters are used but with a "Q" in front of them.

All Qdos devices are supported – mdv, ram, flp, win; no mention is made of fdk. Only floppies are supported with MS-DOS. The working device can be changed easily by typing the new device name, as is normal with MS-DOS. Wild card entries in commands are supported; parameters are entered as with Qdos, according to the conventions given in the QL User Guide. A useful feature is that all wild card commands can be interrupted by pressing ESC. Where choices need to be made, the standard Y/N/A/Q (Yes/No/All/Quit) convention is used, as in *Toolkit II*.

As with XOver and DiscOver, MS-DOS-to-Qdos transfers include the appropriate change to the format; period is changed to underscore and vice versa. If a QL file name is longer than the MS-DOS limit of eight characters before the period and three after it, "surplus" characters will be removed during conversion.

When the program is booted it loads the run-time Turbo Toolkit, which checks the Qdos version of the QL. If that is not JS a window appears, holding a message explaining that SB commands can become duplicated and cause problems when they are called and you are offered the chance to "zap" duplicates. In the case of my system, two of the Turbo commands which are in the boot were branded "bad name" unless the Turbo_TK_code extensions file had been loaded previously; presumably the author did not have this problem but some other users might.

Choosing "Y" once the boot was loaded resulted in several screensfull of duplicates being listed as zapped, because the Turbo_TK_code and run-time Turbo extensions files

have some commands in common. Assuming this function works fully and safely it is most welcome.

The system free memory fell by about 58KB and a short command file was run automatically, changing the "QDOS device" to RAM1__ – but only if a RAM-disc facility is already in the QL – and changing the printer name to PAR if a parallel port exists. You are then presented with a command line prompt looking very like that of MS-DOS – B:/>. Pressing F4 clears the screen, leaving just the command line. F3 writes a list of available commands to the screen, on two lines. In typical current PC menu fashion, pressing a single letter – usually the first – of the command name and then pressing ENTER activates that command.

The start-up device is B: (floppy 2) and initial attempts to get Qdos or MS-DOS directories from that drive were unsuccessful. Trying to change it to A: and C: resulted in a crash, a rare occurrence on my system these days. Re-loading was unsuccessful as it stopped after the surplus extensions had been zapped but a second try was successful. The change to drive A: this time was satisfactory and a directory was obtained of a disc taken from a PC, in floppy 1. The instructions say that drives A: to D: are supported and there was no difficulty when any of these drives was requested subsequently. Likewise there were no subsequent problems obtaining either Qdos or MS-DOS directories from either drive.

Display

If the disc contains any sub-directories the program does not give details of them but it will display files in the root directory. When there are no root files the presence of sub-directories produces the message <Unsupported disc>. There was some uncertainty about what was on the disc used for checking, as only one of nine files copied to it on the PC was identified when a directory was done with MS-QLink. The impression given was that files with some extensions are not recognised but the available space is nevertheless detected correctly.

In the extreme case you can get a <0 files> message when the disc has dozens of files on it. There are other oddities, such as being told there is no Volume Label on a disc when clearly it has one and the available disc space being somewhat different from that quoted on the PC; the latter may be a function of the PC program used for checking space, giving only the actual usable space. The file sizes given were in agreement.

Option

Considering the MS-DOS functions first, the Format command <F> works well and has the useful option of doing either a full format or just deleting the Directory entries for all files. The latter method is usually sufficient, a complete format really being required only when a disc is new – unformatted. The quick format takes less than 10 seconds, which will please most users. The full format takes about the same time as you would expect with Qdos or MS-DOS; 40- and 80-track drives are allowed for.

For example, <F A: 8> will perform an MS-DOS or quick format on the disc in drive A: (floppy 1), treating it as 80-track. With the proviso that files not recognised in the directory operation are not recognised by the other functions either, Directory/Wide – several files displayed on one line – Erase, reName and View work well. Operations are generally fairly fast; in case any reader suspects it there is not the delay associated with using an emulator. Problems are generally indicated by sensible MS-DOS-like error messages.

The Qdos commands are basically the same as the MS-DOS ones, with relatively minor differences for most users. The format is only the full type. File names up to 36 characters long can be displayed. When files are viewed, character codes are not translated, as they may be with MS-DOS files, there being no need to avoid potentially-disruptive codes when the QL is in "native mode".

The Wide directory does not operate, because most of the screen width is occupied by the information on single files – size, type byte, date of recording, file name. Although most files on the test disc were noted, nine out of 25 were not there and there was no obvious pointer from the file names to the difficulty. Again, the available space on the disc was correct.

Several commands allow output to be re-directed to a printer. The irritating inconsistency found with some programs, where single-key responses are sometimes accepted directly and sometimes need to be followed by ENTER, is absent from MS-QLink – a very refreshing feature. The <Silent> command effectively switches other commands from being foreground, question-and-response types, to background, get-on-with-it ones.

When the Silent status is <S 1> you are not shown file names and not asked whether or not you want to over-write existing files and the invoked command – e.g., Erase – assumes you want all files to be dealt with in the same way. That prevents the MS-QLink activities writing to the screen and messing the displays of other jobs. In operations where user intervention is essential – e.g., Format, or when the <Retry or Quit?> message occurs – the screen will be used. It is indicated that the Silent mode is needed with QRam, which attempts to exert control over all screen-writing activities by user programs.

To some extent the use of devices is confusing. On a system which has a RAM disc routine present and two floppy drives, the default selections are floppy 2 (B:) and RAM1__. The latter is acceptable for Qdos but initially I could use only floppy 1 (A:) for MS-DOS operations; later, both drives were used without difficulty. You can put a 3.5in disc in either floppy 1 or 2 and give it a Qdos format but floppy 1 gave a 720KB format and floppy 2 gave 360KB, confirming the impression that a 5.25in drive is expected for floppy 2.

In any event, you can alter the default devices at boot-up time by changing the assignments in the short boot file and they can be re-assigned during operation by typing the new device name. When ram1__ is the default and the cursor prompt is curently A: >, typing <f1p2__Enter> will change the default Qdos device to floppy 2. This is presumably how most users would want it to be, so that files can be copied between MS-DOS and Qdos floppy discs.

The author has tried to cater for whatever possible system configurations the users might have; there might be some initial confusion but, once the desired devices have been set, there should be little difficulty subsequently. In practice, the ability to handle both MS-DOS and Qdos devices from the same command line by using one-letter commands, or their equivalents with a "q" in front, is very helpful. The Last Line Recall feature of the QJump Toolkit still works.

The Read and Write commands deserve more description, as they allow the use of translation files. Command files also are important in reducing the need for repetitive keying. Command and translation files extend the basic functionality of the program considerably. Systems with either Turbo__TK__code or Super Toolkit II installed can be set to call these files from other devices, rather than the default f1p1.

Transferred

The Read command <R> enables an MS-DOS file to be transferred from an MS-DOS disc to a Qdos device and converted to the Qdos format in the process. The Write command <W> accomplishes the reverse process, Qdos to MS-DOS. When the program is started it attempts to load a default (Qdos) translate file. As files can be created using a supplied SB program it is for the user to create a suitable file or files of his requirements. While the familiar keyboard characters can be passed either way between PC and QL without difficulty, control codes and some displayable characters are used differently; for this reason, the translate operation is split into two sections, one handling keyboard codes 32 to 126 and the other control codes 0 to 31 and 127.

There is also the matter of how lines are terminated and the New Line code is set auto-

matically as one or two CR LF (carriage return/line feed) pairs, as appropriate. Other control codes are changed to Spaces. The same translate file can contain entries for both Read and Write operations, the form being very simple; <hmmr> would convert instances of "h" into "m", and "m" into "r".

Success

Files were passed successfully between *text*^{87} and *The Editor* on the QL and *WordPerfect 5.0* on an AT. The only hitch was a consequence of the automatic filename conversion function, which changed the extension <.WP> to <_WP> correctly but added a space on the end, presumably on the assumption that all extensions have three characters in them. The Space is supported by MS-QLink as a valid character, which could be a useful function; in this case the space was not evident until it was located using the View function of *Ice*. The transferred files loaded correctly in their new homes.

Command files take some of the drudgery from repeated operations. They can be created using Quill. Apart from the normal commands mentioned, a command file can contain Echo, Wait and Beep, which respectively display a string on the screen, display a string and wait until you press any key, and display a string and beep continuously until any key is pressed. Obvious uses of command files are to change defaults and name regularly-used files for Read or Write operations.

As the Atari ST disc format is based on the MS-DOS format you can make transfers between ST and QL, provided the Atari drive is double-sided and both QL and ST drives have the same number of tracks (40 or 80).

The Discopy program is an EXECutable background task for copying – not converting – whole MS-DOS, Atari TOS or Qdos discs. The version supplied worked with Qdos discs but halted when presented with MS-DOS discs. As indicated by the author, the program as supplied on the review disc was not ready for distribution but it may have been completed now. Complete discs in either format can be copied with MS-QLink and that would seem a better buy for the extra few pounds but copying in the same format is slower because of the need to copy from disc to RAM and back to disc, changing the default disc drive in-between.

Programs from relatively-unknown sources tend to put me in a somewhat negative frame of mind from the start, as I wonder how many things will go wrong in the review period. This is not to say that programs from the better-known sources are trouble-free – they rarely are – but the programmer with less experience can easily overlook common difficulties which undoubtedly will occur once a program gets into the hands of ordinary users.

The lengthy process of combining all the instruction files and reducing the length of the paper output gave me time to consider the words before pressing the keys and the impression gained from the instructions was all favourable. It was only when the program was first run that doubts appeared.

There are some weaknesses in the program structure and it takes a fair amount of time to get to grips with the way devices are handled but the overall impression is that this is not another cheap and insignificant amateur program. There are real, useful – and usable – functions in it which are suitable for the user who has both PC and QL and needs to transfer files between them but does not want to spend much money.

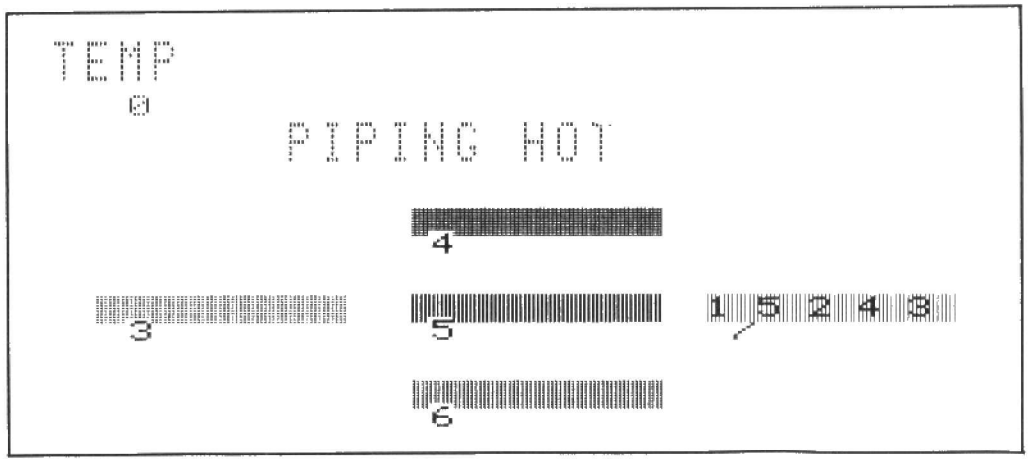
The low price in no way indicates low quality. As the program has been developed over several years it is likely that improvements have been made since the review copy was despatched but the experienced QL user might not find the flaws in the tested version big enough to cause undue difficulty. Even so, it is suggested that potential buyers ask first if the problem of recognising certain file types has now been rectified.

THE P+R:O=G<S

If you have a program worthy of consideration, send it to 'The Progs', Sinclair QL World, Greencoat House, Francis Street, London SW1P 1DG. We pay for everything published at the usual page rates.

PIPING HOT

by Bill Currie



'Piping Hot' was developed out of a need to use a CLEAR instruction while retaining certain subsidiary routines (for example, for QLink, for which refer to Simon Goodwin's article in *QL World*, October 1988). Difficulty had been experienced in the repeated use of Qdos aRIthmetic routines in developing a program on the Mandelbrot set, and it was found that intermittent use of the CLEAR instruction helped to over come these.

Once opened, two pipes can be easily joined, using the procedure j(chana,chanb) in a couple of lines of programming. Information may then be PRINTed to chana, a CLEAR instruction issued and the information retrieved by INPUT to chanb.

The program is in two parts: the first part is a series of utilities. These are:

WINdows for 80 character width

Hex converter

Channel definition data and pipe contents

Link (two way 'END to END')

Join (one way only)

Separation of pipes

iT$emise contents of a pipe

Remaining useable iTT$ems in a pipe

FLow to move data from the first to the second pipe — not automatic

The second part of the program is a straightforward game/experimentor to illustrate possible uses of pipes, for instance in sort routines as well as with CLEAR instructions. If the program is run it will go straight into the game.

Readers may like to note that if the program is LOADed, typing, for example, 'c 1' will cause a printout of channel definition data. It is almost as if each independent procedure were a separate program. This idea can be very usefully developed, particularly with short (for example, one-letter) procedure names.

Continued on Page 44

```
100 game
110 REMark    ********     Queue pLaying the Pipes        ********
120                              :
130 REMark        (c) Bill Currie (Oct 1989)
140                              :
150 REMark Care must be taken to open channels 'in sequence'
160                              :
170 REMark By entering the experimental mode the game may, by choosing suitable
    objectives, be made into a two or (multi) player (gambling) game.
180                              :
190 REMark ** make screens(mode4) of 80 character width (printer compatible) **
200                              :
210 DEFine PROCedure wins
220 WINDOW#1,484,200,15,16:PAPER 0:INK 7:CLS#1
230 WINDOW#2,484,200,15,16:BORDER#2,1,2:PAPER#2,0:INK#2,7:CLS#2
240 WINDOW#0,480,40,17,216:PAPER#0,0:INK#0,7:CLS#0
250 END DEFine
260                              :
270 REMark                *** hex converter ***
280                              :
290 DEFine FuNction h(x$)
300 LOCal i,k,v,val
310       :
320 REMark Write eg $fA6 as h('fA6')
330       :
340 val=0
350 FOR i=1 TO LEN(x$)
360 k=CODE(x$(i))
```

Continued from Page 42

```
370 IF k>47 AND k<58 THEN v=k-48
380 IF k>64 AND k<71 THEN v=k-55
390 IF k>96 AND k<103 THEN v=k-87
400 val=v+16*val
410 END FOR i
420 RETurn val
430 END DEFine
440                                 :
450 REMark ******************** PIPE DATA **********************
460                                 :
470 REMark   Utility to inspect pipe and channel data
480 REMark eg if a pipe is created by OPEN#3,PIPE_50 then [enter] c 3
490                                 :
500 DEFine PROCedure c(ch)
510 MODE 4
520 wins
530 x = PEEK_L(PEEK_L(h('28000')+h('78'))+4*ch)
540 PRINT x!!'address of channel definition block'
550 PRINT PEEK_L(x)!!'CH.LEN (length of the definition block for the #)'
560 PRINT PEEK_L(x+4)!!'CH.DRIVR (address of # driver)'
570 PRINT PEEK_L(x+8)!!'CH.OWNER (ID of # owner)'
580 PRINT PEEK_L(x+12)!!'CH.RFLAG (address set when # closed)'
590 PRINT PEEK_W(x+16)!!'CH.TAG (tag allocated from opening sequence,not #no)'
600 PRINT PEEK(x+18)!!'CH.STATus:0 OK,-1 A1 absolute,$80 A1 relative to A6,-ve w
aiting'
610 PRINT PEEK(x+19)!!'CH.ACioTN (for waiting job, value for D0)'
620 PRINT PEEK(x+20)!!'CH.JOBWT (ID of Job waiting on IO)'
630 PRINT
640 PRINT PEEK_L(x+24)!!'CH.QIN (pointer to input queue (if a pipe))'
650 PRINT PEEK_L(x+28)!!'CH.QOUT (pointer to output queue (if a pipe))'
660 PRINT PEEK_L(x+32)!!'link to next queue (if a pipe)'
670 PRINT PEEK_L(x+36)!!'pointer to end of queue (if a pipe)'
680 PRINT PEEK_L(x+40)!!'pointer to insert location (if a pipe)'
690 PRINT PEEK_L(x+44)!!'pointer to retrieve location (if a pipe)'
700 PRINT
710 CLS#0:INPUT#0,'PIPE contents in code (with pointers) ? y/n ';ap$
720 IF ap$=='y' THEN INK 4:PRINT 'Pipe contents in ASCII code,4 pointers,1st ite
m at end,2nd at start ':INK 7
730 cs=PEEK_L(x+28):cf=PEEK_L(x+36)
740 IF ap$=='y' THEN FOR k=cs TO cf:PRINT PEEK(k)!;
750 CLS#0:INPUT#0,'PIPE contents in character form (no pointers) ? y/n ';ap$
760 IF ap$=='y' THEN INK 4:PRINT \'Pipe contents,no pointers,in character form':
INK 7
770 IF ap$=='y' THEN FOR k=cs+16 TO cf:IF PEEK(k)>=32 AND PEEK(k)<=191 THEN PRIN
T CHR$(PEEK(k))!;:ELSE IF PEEK(k)=0 THEN PRINT CHR$(48)!;:ELSE IF PEEK(k)=10 THE
N PRINT CHR$(92)!;:ELSE PRINT CHR$(183)!;:END FOR k
780 END DEFine
790                                 :
800 REMark ************************************************************
810                                 :
820 REMark       *****  The Main (V Simple) Utilities  *****
830                                 :
840 REMark                      Two Way Link
850                                 :
860 REMark  This will link two pipes both ways(end to end); eg try
            open#3,pipe_100:open#4,pipe_100
            print#3,'message1'\'message2':print#4,777\463
                        1  3,4
            input#4,a$:print a$
            input#3,b$:print b$
870                                 :
880 DEFine PROCedure l(cha,chb)
890 d=PEEK_L(PEEK_L(163960)+4*cha):e=PEEK_L(PEEK_L(163960)+4*chb)
900 POKE_L (d+32),PEEK_L(e+28):POKE_L(e+24),PEEK_L(d+28)
910 POKE_L(e+32),PEEK_L(d+28):POKE_L(d+24),PEEK_L(e+28)
920 END DEFine
930                                 :
940 REMark        A procedure for a one way link ,ex j 3,4
950                                 :
960 DEFine PROCedure j(chana,chanb)
```

```
970 d=PEEK_L(PEEK_L(163960)+4*chana):e=PEEK_L(PEEK_L(163960)+4*chanb)
980 POKE_L (d+32),PEEK_L(e+28):POKE_L(e+24),PEEK_L(d+28)
990 END DEFine
1000                                    :
1010 REMark     A procedure to separate two pipes :eg     s 4,3
1020                                    :
1030 DEFine PROCedure s(chana,chanb)
1040 d=PEEK_L(PEEK_L(163960)+4*chana):e=PEEK_L(PEEK_L(163960)+4*chanb)
1050 POKE_L (d+32),0:POKE_L(e+24),0
1060 END DEFine
1070                                    :
1080 REMark ************************************************************
1090                                    :
1100 REMark        A function to investigate the contents of a pipe
          eg print t$(3,2) will print 2nd item in a channel 3 pipe
       NB items remain in the pipe even after use, only the pointers update
1110                                    :
1120 DEFine FuNction t$(chana,posn)
1130 d=PEEK_L(PEEK_L(163960)+4*chana)
1140       :
1150 IF posn=1 THEN
1160 e=PEEK_L(d+36)
1170 pk=PEEK(e-1)
1180 IF pk=0 THEN RETurn "pipe "&chana&" is empty"
1190 IF pk=10 THEN RETurn "."
1200 a$=CHR$(pk)
1210 f=PEEK_L(d+28)
1220 k=0
1230 REPeat looop
1240 pk=PEEK(f+16+k)
1250 IF pk=10 THEN EXIT looop
1260 a$=a$&CHR$(pk)
1270 k=k+1
1280 END REPeat looop
1290 RETurn a$
1300 END IF
1310       :
1320 IF posn>1 THEN
1330 f=PEEK_L(d+28)
1340 k=0:r=1
1350 REPeat loop
1360 pk=PEEK(f+16+k)
1370 IF pk=0 THEN RETurn "unfilled"
1380 IF pk=10 THEN r=r+1
1390 k=k+1
1400 IF r=posn THEN EXIT loop
1410 END REPeat loop
1420 ks=k
1430 REPeat lp
1440 pk=PEEK(f+16+k)
1450 IF pk=0 THEN RETurn "unfilled"
1460 IF pk=10 THEN EXIT lp
1470 k=k+1
1480 END REPeat lp
1490 ke=k-1
1500 a$=""
1510 IF ke>=ks THEN
1520  FOR k=ks TO ke:a$=a$&CHR$(PEEK(f+16+k))
1530 END IF
1540 RETurn a$
1550 END DEFine
1560                                    :
1570 ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
1580                                    :
1590 REMark  A function to investigate the remaining usable items in a pipe
        eg print tt$(4,5) will print the 5th item remaining in channel 4
1600                                    :
1610 DEFine FuNction tt$(chana,posn)
1620 d=PEEK_L(PEEK_L(163960)+4*chana)
```

```
1630 IF PEEK_L(d+44)=((PEEK_L(d+36))-1) THEN RETurn t$(chana,posn)
1640 f=PEEK_L(d+44)
1650 IF posn=1 THEN
1660    IF f=0 THEN RETurn "unfilled"
1670    END IF
1680    IF f=10 THEN RETurn ""
1690    END IF
1700 END IF
1710 k=0:r=0:ks=0
1720 REPeat llp
1730 pk=PEEK(f+k)
1740 IF pk=0 THEN RETurn "unfilled"
1750 IF pk=10 THEN
1760    r=r+1
1770    IF r=posn THEN
1780    ke=k-1
1790    ELSE
1800    ks=k+1
1810    END IF
1820 END IF
1830 k=k+1
1840 IF r=posn THEN EXIT llp
1850 END REPeat llp
1860 a$=""
1870 FOR k=ks TO ke:a$=a$&CHR$(PEEK(f+k))
1880 RETurn a$
1890 END DEFine
1900 STOP
1910                                :
1920 :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
1930                                :
1940 REMark Simple experimentation reveals that pipes do not quite behave
  as expected. In particular it would be expected that flow from one pipe
  to the next would move data usably into the next pipe. This is achieved
                        by the next procedure.
1950                                :
1960 DEFine PROCedure f1(chana,chanb)
1970 INPUT#chanb,aa$:PRINT#chanb,aa$
1980 END DEFine
1990                                :
2000 :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2010                                :
2020 REMark A major facility is the avoidance of data loss by clear commands -
                      type in   dem  [enter]
2030                                :
2040 DEFine PROCedure dem
2050 CLOSE#3:CLOSE#4
2060 OPEN#3,pipe_100:OPEN#4,pipe_100
2070 j 3,4
2080 a$='demonstration'
2090 PRINT#3,a$
2100 CLEAR        :REMark a$ would normally be lost by this instruction
2110 PRINT a$     :REMark This should show as *
2120 INPUT#4,a$
2130 PRINT a$     :REMark a$ is retrieved
2140 STOP
2150 END DEFine
2160                  :::::::::::::::::::::::::::::::::
2170 :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2180                                :
2190 REMark               Piping Hot (Game Program)
2200                                :
2210 DEFine PROCedure game
2220                                :
2230 wins:MODE 8:CLS:CLS#0
2240 OPEN#3,pipe_100:OPEN#4,pipe_100:OPEN#5,pipe_100:OPEN#6,pipe_100:OPEN#7,pipe
_100
2250 CLEAR
2260 CLS
```

```
2270 instructions
2280 screen
2290 startup
2300 play
2310                              :
2320 DEFine PROCedure instructions
2330 CLS#0:INK 7
2340 PRINT#0,'Use  valid  instructions  only  to'\' arrange numbers in order in
pipe 3'
2350 PRINT#0,'  Press v for valid instructions'
2360 a$=INKEY$(-1)
2370 MODE 4
2380 PAPER 0:CLS
2390 PRINT \'              Examples of valid instructions :'\\\\
2400 PRINT '             j 3,4  [to join two pipes]'
2410 PRINT '             l 5,3  [to link two pipes end to end]'
2420 PRINT '             s 6,7  [to separate two pipes]'
2430 PRINT '             fl 6,7  [flow of one item]'
2440 PRINT '             v 3   [to inspect remaining contents of pipe]'
2450 PRINT '             E  [to enter experimental mode - disable temperature]'
2460 PRINT '             G  [to re-enter game mode]'
2470 PRINT \\\'              PRESS ANY KEY - h will keep pipes hidden'
2480 spec$=INKEY$(-1)
2490 END DEFine
2500                              :
2510 DEFine PROCedure screen
2520 MODE 8
2530 BORDER#1,2,4
2540 PAPER 0:CLS:OVER -1:INK 2,6,1
2550 CSIZE 3,1
2560 PRINT \\'       PIPING HOT'
2570 AT 0,0:PRINT 'TEMP'
2580 CSIZE 0,0
2590 OVER 0
2600 INK 6:AT 2,3:PRINT 0
2610 OPEN#13,scr_120x10a40x120:PAPER#13,2:CLS#13
2620 OPEN#15,scr_120x10a190x120:PAPER#15,6:CLS#15
2630 OPEN#16,scr_120x10a190x150:PAPER#16,5:CLS#16
2640 OPEN#14,scr_120x10a190x90:PAPER#14,3:CLS#14
2650 OPEN#17,scr_120x10a330x120:PAPER#17,1:CLS#17:INK#17,7
2660 PAPER 0:INK 7
2670 AT 11,3:PRINT 3
2680 AT 11,15:PRINT 5
2690 AT 11,27:PRINT 7
2700 AT 14,15:PRINT 6
2710 AT 8,15:PRINT 4
2720 END DEFine
2730                              :
2740 DEFine PROCedure startup
2750 a$='12345'
2760 b$=''
2770 FOR i=5 TO 1 STEP -1
2780 k=INT(RND(i-1))+1
2790 b$=b$&a$(k)
2800 IF i=1 THEN GO TO 2860
2810 SELect ON k
2820 =1:a$=a$(2 TO i)
2830 =i:a$=a$(1 TO k-1)
2840 =REMAINDER :a$=a$(1 TO k-1)&a$(k+1 TO i)
2850 END SELect
2860 END FOR i
2870 FOR i=1 TO 5:PRINT#7,b$(i):PRINT#17,b$(i)!;
2880 END DEFine
2890                    :
2900 DEFine PROCedure pr(chn,pos)
2910 PRINT #(chn+10),tt$(chn,pos)!;
2920 END DEFine
2930                    :
2940 DEFine PROCedure v(chan)
```

```
2950 CLS#(chan+10)
2960 INK#(chan+10),0:INK#17,7:INK#13,7
2970 n=0
2980 REPeat looping
2990 n=n+1
3000 ee$=tt$(chan,n)
3010 IF CODE(ee$(1))>CODE('5') OR CODE(ee$(1))<CODE('1') THEN EXIT looping
3020 PRINT #(chan+10),ee$!;
3030 END REPeat looping
3040 END DEFine
3050                                :
3060 DEFine PROCedure play
3070 temperature=0:inc=0:INK 6:marker = 0:dis$='0'
3080 DIM check$(7,7)
3090 FOR i=1 TO 7
3100 FOR k=1 TO 7
3110 check$(i,k)="0"
3120 END FOR k
3130 END FOR i
3140                    :
3150 REPeat loping
3160                                :
3170 CLS#0:INPUT#0,'Instruction ?  E(xperiment) G(ame)'\'eg l 6,5 : j 7,4 : s 3,
4 : f1 7,4 : v 6'\ins$
3180                                :
3190 IF ins$=='e' THEN dis$='1':END REPeat loping
3200 IF ins$=='g' THEN dis$='0':END REPeat loping
3210                                :
3220 IF LEN(ins$)<3 OR LEN(ins$)>6 THEN PRINT#0,'invalid':PAUSE:END REPeat lopin
g
3230 IF ins$(1)INSTR"jlsfvJLSFV"=0 THEN PRINT#0,'invalid':PAUSE:END REPeat lopin
g
3240                                :
3250 IF ins$(1)=='v' AND LEN(ins$)<>3 THEN PRINT#0,'invalid':PAUSE:END REPeat lo
ping
3260 IF ins$(1)=='f' AND LEN(ins$)<>6 THEN PRINT#0,'invalid':PAUSE:END REPeat lo
ping
3270 IF ins$(1)=='f' AND (ins$(2)<>'l' AND ins$(2)<>'L') THEN PRINT#0,'invalid':
PAUSE:END REPeat loping
3280 IF ins$(1)INSTR'jlsJLS'<>0 AND LEN(ins$)<>5 THEN PRINT#0,'invalid':PAUSE:EN
D REPeat loping
3290                                :
3300 IF ins$(1)INSTR"jlsJLS"<>0 AND ins$(3)INSTR"34567"=0 THEN PRINT#0,'invalid'
:PAUSE:END REPeat loping
3310 IF ins$(1)INSTR'jlsJLS'<>0 THEN IF ins$(5)INSTR"34567"=0 THEN PRINT#0,'inva
lid':PAUSE:END REPeat loping
3320 IF ins$(1)=="j" THEN FOR i=3 TO 7:check$(ins$(3),i)='0':check$(i,ins$(5))="
0":END FOR i : check$(ins$(3),ins$(5))="1"
3330 IF ins$(1)=="l" THEN FOR i=3 TO 7:check$(ins$(3),i)=0:check$(ins$(5),i)=0:c
heck$(i,ins$(3))=0:check$(i,ins$(5))=0:END FOR i:check$(ins$(3),ins$(5))="1":che
ck$(ins$(5),ins$(3))="1"
3340 IF ins$(1)=="s" THEN check$(ins$(3),ins$(5))="0"
3350 IF ins$(1)=='j' THEN j ins$(3),ins$(5):inc=ins$(5)*2
3360 IF ins$(1)=='l' THEN l ins$(3),ins$(5):inc=ins$(5)*2
3370 IF ins$(1)=='s' THEN s ins$(3),ins$(5):inc=ins$(5)*2
3380 IF ins$(1)=='f' THEN IF ins$(4)INSTR'34567'=0 THEN PRINT#0,'invalid':PAUSE:
END REPeat loping
3390 IF ins$(1)=='f' THEN IF ins$(6)INSTR'34567'=0 THEN PRINT#0,'invalid':PAUSE:
END REPeat loping
3400 IF ins$(1)=='f' THEN IF check$(ins$(4),ins$(6))="0"  THEN PRINT#0,'invalid'
:PAUSE:END REPeat loping
3410 IF ins$(1)=='f' THEN IF tt$(ins$(4),1)INSTR'12345'=0 THEN PRINT#0,'invalid'
:PAUSE:END REPeat loping
3420 IF ins$(1)=='f' THEN f1 ins$(4),ins$(6):inc=(ins$(6))*2:IF spec$<>'h' AND s
pec$<>'H' THEN v ins$(4):v ins$(6)
3430 IF ins$(1)=='v' THEN IF ins$(3)INSTR"34567"=0 THEN PRINT#3,'invalid':PAUSE:
END REPeat loping
3440 IF ins$(1)=='v' THEN v ins$(3):inc=10
```

```
3450                    :
3460 IF dis$='1' THEN inc=0
3470 temperature=temperature+inc
3480 IF spec$=='h' THEN temperature=temperature-1
3490 AT 2,3: PRINT temperature!!!
3500 IF temperature>100 THEN GO TO 3550
3510 IF dis$='0' AND tt$(3,1)='1' AND tt$(3,2)='2' AND tt$(3,3)='3' AND tt$(3,4)
='4' AND tt$(3,5)='5' THEN marker =1:GO TO 3540
3520 END REPeat loping
3530                    :
3540 IF marker=1 THEN CLS:CLS#0::PRINT \\' CONGRATULATIONS You won':GO TO 3560
3550 CLS:CLS#0:PRINT \\'    overheating'\\\'      you  lost'
3560 tune
3570 PRINT \\'press g for another go'\'or s to stop'
3580 a$=INKEY$(-1)
3590 IF a$=='g' THEN game
3600 IF a$=='s' THEN STOP
3610 GO TO 3580
3620                    :
3630 END DEFine
3640                              :
3650 END DEFine
3660                              :
3670 DEFine PROCedure tune
3680 RESTORE
3690 DIM a(49)
3700 k1=INT(RND(7)):FOR k=1 TO 49:READ a(k):BEEP 9500,a(k),a(k)-5,2000,4,6,0,0:P
AUSE 18:INK ((k+k1) MOD 5)+2:CIRCLE 120,30,10+(k MOD (1+INT(RND(6))))
3710 DATA 52,48,48,41,36,36,41,48,48,59,68,68,52,48,48,48,48,36,28,24,24,24,28
3720 DATA 28,28,24,24,28,36,36,28,33,36,41,48,59,68,52,48,48,36,28,24,28,36,41,4
1,41,48
3730 END DEFine
3740                         ::::::::::::::::
3750 REMark             ::::::      End      :::::::
3760                    ::::::::::::::::::::::::::::::
```

PDQL Computer Systems and Software

UNIT 1, HEATON HOUSE CAMDEN STREET BIRMINGHAM B1 3BZ 021 200 2313

MAKE THE MOST OF YOUR QL AND THOR

for home and business use — Order from list or write for current catalogue. Prices include VAT and postage within the UK. Cheques with orders to, and full descriptive list available from PDQL. Please state details of toolkit, extended memory and or disc.

SPRING INTO ACTION WITH PDQ SOFTWARE

BUSINESS PROGRAMS:

PDQ-Payroll **£80.00**

Trading Accounts (inc. Analyser)
The complete accounting system **125.00**

Cash Trader Upgrade **£85.00**
(For use with sub 33 sales and purchase ledger accounts)

Analyser **£25.00**

Mailmerge de Luxe **£20.00**

IMPROVE YOUR ARCHIVE

PDQL is offering a special ARCHIVE package -

ARCHIVE TUTOR - which teaches you what you didn't know about Archive

SEDIT - the sensible way to build your Archive screen

RECOVER - the essential insurance program to cope with lost database files

All three for £45 (You can still buy them separately!)

UTILITY PACKAGE

COMPARE - the easy way to detect differences (even an extra space) in two files supposedly the same. Contains the famous magic sliding panel. Suitable for any text/system/SuperBASIC file - easy to use alignment/reposition options - ignore first n columns (e.g. line numbers).

PDQ-COPY - not merely a fast copier. You can obtain a complete (or exceptions only) listing of files on different media (e.g. disc or cartridge), copy dates, sizes. You can copy all, by Yes/No input, by up to three strings e.g. all _doc files beginning PDQL, optional pause before overwriting, format destination disc/cartridge or delete files; set it up, pour your coffee, and PDQ-COPY has finished its task.

TexTIDY - tidies up your _doc files by converting to plain text; OR to Wordstar or DOS Quill format prior to using DiscOVER; converts from DiscOVERed DOS Quill to QL Quill format; tidies DiscOVERed Wordstar or other text files prior to importing into EDITOR or importing into Quill. In addition you can import into Quill your SuperBASIC program, edit, save, TexTIDY and BINGO! run or compile the TexTIDied file.

All three for £25 (You can still buy them separately!)

Access VISA

UNIT 1, HEATON HOUSE
CAMDEN STREET
BIRMINGHAM B1 3BZ
021 200 2313

Your ticket to DiscOVER

QL to IBM	RETURN
FIRST CLASS	Any Day £29.50

892138 092138

DiscOVER is the essential program for transferring any file EITHER WAY between QL and IBM format. NO CABLES NEEDED. NO SIDE-by-SIDE MACHINES. Run DiscOVER, select to or from the QL, transfer all or cursor selected files. Features include optional symbol translate; delete file; automatic file-name change for IBM/QL compatibility. Available on 3.5 or 5.25 disc

PDQL have a larger version including CPM and BBC transfer facilities as well as IBM - Multi-DiscOVER

£39.00

European orders may be placed with:
DANSOFT, Raadhusstraede 4B1, 1466 Copenhagen K, Denmark. Tel 45(01) 930347 11.